资深SAP专家扛鼎之作
为SAP ABAP开发提供解决方案

孙东文 郭欢 张岩 等编著

SAP ABAP开发详解与高端应用

SAP
ABAP
Detailed Development Technology
- High-end Application







机械工业出版社
CHINA MACHINE PRESS

VISIT…

LANZAROTE Caliente.com

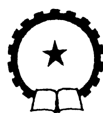
作者简介



孙东文，笔名：东方先生，SAP顾问，山东沾化人。毕业于大连舰艇学院并获得学士学位，从事SAP推广、实施、维护及教学近10年。

SAP ABAP 开发详解与高端应用

孙东文　郭　欢　张　岩　等编著



机 械 工 业 出 版 社

本书主要介绍了 ABAP 面向对象编程、ALV 列表、接口、增强及 ABAP 开发人员求职面试的相关技术问题。ABAP 面向对象编程部分由浅入深地讲解了面向对象的相关概念，并从实际应用出发，举例说明了 ABAP 编程所涉及的本地对象和全局对象的编辑实现及应用效果；ALV 列表部分全面地介绍了 ALV 的种类以及各种 ALV 的实现方式，并详述了面向对象 ALV 列表及 ALV 树形列表的编辑实现及应用效果；接口部分系统地介绍了各类接口的原理及实现方法，并详述了 IDOC、RFC、BAPI 的相关概念及原理；增强部分对增强的升级及不同时期的各代增强的原理、查找办法、实现方式作了详细阐述，并举例说明了每一代增强的编辑实现及应用效果；ABAP 面试问题及答案部分提供了较为全面的面试问题集，为求职者打开方便之门。

本书提供了大量配套资源及实例源码，深入剖析了 SAP NetWeaver 架构的关键技术，是 Java 等开发人员深入学习 SAP 系统的必备指南，还适用于 SAP 技术人员和 Java\.NET 平台下 SAP 接口程序的开发人员。

图书在版编目（CIP）数据

SAP ABAP 开发详解与高端应用 / 孙东文等编著. —北京：机械工业出版社，2015.6

ISBN 978-7-111-50126-8

Ⅰ. ①S… Ⅱ. ①孙… Ⅲ. ①企业管理－应用软件－软件开发
Ⅳ. ①F270.7

中国版本图书馆 CIP 数据核字（2015）第 114323 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）
策划编辑：丁　诚　　责任校对：张艳霞
责任编辑：丁　诚　周　萌
责任印制：李　洋

三河市宏达印刷有限公司印刷

2015 年 7 月第 1 版・第 1 次印刷
184mm×260mm・23.5 印张・579 千字
0001－3000 册
标准书号：ISBN 978-7-111-50126-8
定价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务	网络服务
服务咨询热线：（010）88361066	机 工 官 网：www.cmpbook.com
读者购书热线：（010）68326294	机 工 官 博：weibo.com/cmp1952
（010）88379203	教育服务网：www.cmpedu.com
封面无防伪标均为盗版	金 书 网：www.golden-book.com

前　言

随着 SAP 软件在中国企业的广泛应用，SAP 实施及运维的 IT 服务行业也异军突起，ASP 相关技术也得到了广泛的应用，这其中不乏 SAP 开发的高端技术。

作者根据 SAP 官方教材和项目经验对 ABAP 开发技术做了一下分类，将 ABAP 面向对象编程、ALV 的面向对象应用、接口和增强归作为高级应用技术纳入本书。

以往的书籍仅对 ABAP 高级应用部分做原理剖析，能够提及对面向对象、接口及增强实现的图书是少之又少。本书特点在于对各技术原理、实施前提、实施步骤、实施结果作了详尽的说明，更有 ABAP 求职者相关基础技术及高端技术的面试题，以帮助那些做技术提升的 SAP 开发人员。

学习 SAP 高级开发技术，开发者不仅需要提高自身水平（例如，从面向过程的编程到面向对象编程的转变，从单一开发语言的编程到不同语言之间编程的转变及其之间数据关系映射的了解），也需要熟悉 SAP 相关操作（例如，远程 RFC 的 TCP 链接的创建和调试），更需要 SAP 开发者对各种技术实施后最终体现的结果的认同（例如，如何应用创建的 RFC 链接，高端技术实施应用以后有什么样的效果，是否和需求相吻合等）。

参与本书编写的有孙东文（笔名：东方先生）、郭娟、郭欢、张岩，由于水平有限，书中难免有疏漏和不足，请读者批评指正。

编　者

目　　录

前言
第 1 章　ABAP 面向对象编程 ········ 1
1.1　面向对象的基本要素 ········ 2
1.1.1　封装 ········ 2
1.1.2　多态 ········ 3
1.1.3　继承 ········ 3
1.1.4　面向对象的成员 ········ 4
1.1.5　接口 ········ 10
1.2　本地类的实现 ········ 12
1.2.1　程序设计 ········ 12
1.2.2　程序测试 ········ 26
1.3　全局类的实现 ········ 26
1.3.1　接口制作 ········ 26
1.3.2　接口的使用（类的创建） ········ 31
1.3.3　类的测试 ········ 37
第 2 章　ALV 应用 ········ 38
2.1　函数 ALV 和面向对象事件 ········ 44
2.1.1　程序设计 ········ 44
2.1.2　程序测试 ········ 59
2.2　面向对象 ALV ········ 60
2.2.1　Dynpro 控件 ALV ········ 60
2.3　面向对象 ALV ········ 82
2.3.1　程序设计 ········ 82
2.3.2　程序测试 ········ 95
2.4　ALV 树 ········ 97
2.4.1　程序设计 ········ 97
2.4.2　程序测试 ········ 109
第 3 章　接口 ········ 111
3.1　SAP 数据交换接口的类型 ········ 111
3.1.1　CPI-C 简介 ········ 111
3.1.2　RFC 简介 ········ 112
3.1.3　ALE 简介 ········ 112
3.2　SAP R/3 的接口方式 ········ 114
3.2.1　IDOC 接口 ········ 114
3.2.2　RFC 接口 ········ 119

3.2.3 BAPI 接口 ······ 130
3.3 ALE/IDoc 应用 ······ 135
3.3.1 SAP 系统间数据传输配置 ······ 135
3.3.2 SAP 系统间数据扩张传输 ······ 149
3.3.3 SAP 系统与其他系统间数据传输 ······ 186
3.4 RFC 应用 ······ 187
3.4.1 同步 RFC 方式的远程函数调用 ······ 187
3.4.2 异步 RFC 方式的远程函数调用 ······ 188
3.4.3 并行 RFC 方式的远程函数调用 ······ 190
3.4.4 事务 RFC 方式的远程函数调用 ······ 193
3.4.5 队列 RFC 方式的远程函数调用 ······ 196
3.4.6 外部系统 RFC 方式的远程调用 ······ 198
3.5 BAPI 的应用 ······ 217
3.5.1 SAP 业务对象 ······ 217
3.5.2 BAPI 定义 ······ 218
3.5.3 BAPI 调用 ······ 225
第 4 章 增强 ······ 236
4.1 概述 ······ 236
4.1.1 第一代增强 ······ 236
4.1.2 第二代增强 ······ 238
4.1.3 第三代增强 ······ 243
4.1.4 第四代增强 ······ 245
4.2 第一代增强的应用 ······ 245
4.3 第二代增强的应用 ······ 245
4.3.1 数据元素及关键字 ······ 245
4.3.2 字段增强 ······ 246
4.3.3 GuiXT 增强 ······ 250
4.3.4 表增强 ······ 255
4.3.5 函数增强 ······ 259
4.3.6 菜单增强 ······ 263
4.3.7 屏幕增强 ······ 268
4.4 第三代增强的应用 ······ 276
4.4.1 BADI 增强 ······ 276
4.4.2 BTE 增强 ······ 287
4.5 第四代增强（Code Enhancement）的应用 ······ 293
第 5 章 ABAP 面试问题及答案 ······ 298
5.1 语法 ······ 298
5.1.1 基础知识 ······ 298
5.1.2 应用技术 ······ 301

5.2 数据字典相关问题 304
5.3 报表相关问题 306
5.3.1 基本报表 306
5.3.2 ALV 报表 308
5.3.3 SmartForm 报表 308
5.4 对话程序 309
5.4.1 事件 309
5.4.2 控件 309
5.4.3 语法 310
5.5 批处理 310
5.5.1 原理 310
5.5.2 语法 311
5.6 增强 312
5.7 接口 313
5.7.1 RFC 接口 314
5.7.2 BAPI 接口 315
5.7.3 其他 316
5.8 数据库知识 317
5.8.1 基础知识 317
5.8.2 ABAP 和数据库 318
5.9 业务知识 321
5.10 其他 324
第 6 章 常用工具 328
6.1 SAP 简单报表生成工具 328
6.1.1 Quick View 328
6.1.2 Quick Query 331
6.2 SAP 快速录入工具 345

第 1 章　ABAP面向对象编程

SAP 系统从 4.0 版本开始对 ABAP 语言进行了扩充，增加了 ABAP 类和对象功能。也就是说，可以使用类的所有特性，如封装、多态、接口和继承等，使复杂的应用变得更容易控制且简单化，并且 SAP 要求 ABAP 语言严格向上兼容。

ABAP 类语言跟其他面向对象的语言一样，有类和接口，类又包括属性、方法和事件。

一、面向对象

■ 对象（Object）

对象是包含部件并提供服务的程序代码段，其中部件部分代表该对象的属性，而所提供的服务则被称为方法。

■ 类

类是对象的定义，通过属性和方法的封装来描绘对象。类是对象的模板，相应地，一个对象的类型与它的类相同。一个类是一个对象的抽象描述，它是构建一个对象所需要的一系列指令。对象的属性和方法由类的部件定义，类描述了对象的框架和行为。

■ 类和对象的关系

从技术角度看，对象是程序中类的实例，在运行环境中根据类的定义生成。类和对象是一对多的关系，即可以创建某个类的多个对象，而每个对象均是独立的，拥有代表自身状态的数据值集。

■ 全局类和本地类

全局类是在 Class Builder（事务码：SE24）中定义的类，其被存储在类池中的程序类型为接口池（J）和类别库（K）定义在 Include 程序中的本地类也可作为全局类。所有 ABAP 程序都可以访问并使用全局类。本地类是在 ABAP 程序中定义的类，它和本地接口只能用于定义本地类的程序中，应用时系统首先查找指定名字的本地类，如果没有找到，再查找全局类。除了有效性问题，全局类和本地类在使用时没有区别。

二、名词解释

1．抽象（abstraction）

抽象是指对于一个过程或者一件制品的某些细节有目的地隐藏，以便把其他方面、细节或者结构表达得更加清楚。

抽象是控制复杂性时最重要的工具。人们通常使用一些简单的工具来建立、理解和管理复杂的系统。其中最重要的技术称为“抽象”。

2．封装

封装就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。

3．继承

继承可以使用现有类的所有功能，并在无需重新编写原有类的情况下对这些功能进行扩展。

4．多态

多态允许将父对象设置成为与一个或多个其子对象相等的技术，赋值之后，父对象就可以根据当前赋值使它的子对象以不同的方式运作。简单地说，就是允许将子类类型的指针赋值给父类类型的指针。

1.1 面向对象的基本要素

1.1.1 封装

对象可以对其内部资源的可见性进行限定，每个对象都有一个接口，决定了其他对象如何与之交互。

基类包括多个派生类，但该基类是作为模板出现，并不需要有任何对象作为实例。

本地类由 ABAP 源代码组成，封装在 CLASS ... ENDCLASS 中。一个完整的类定义包括声明部和执行部，声明中的<class>部分是一个语句块。

语法：

```
CLASS <class> DEFINITION.
   ...
ENDCLASS.
```

它包含类所有部件（属性、方法、事件）的声明。当定义一个本地类时，声明部分属于全局 Program 数据，所以必须把它放在 Program 的开始。

如果在类的声明部声明方法，必须也为它写一个执行部。这是由更多的语句块组成的。

语法：

```
CLASS <class> IMPLEMENTATION.
   ...
ENDCLASS.
```

在类的执行部分包含了所有类方法的执行，本地类的执行部是一个处理块。那些不在一个处理块中的后面的代码是不能访问的。

1．声明为抽象类（Abstract Class）

语法：

```
CLASS <class> DEFINTION ABSTRACT.
   METHODS meth ABSTRACT...
ENDCLASS.
```

不可用 CREATE OBJECT 语句创建类对象，只作为派生类模板；抽象方法不可在类本身实现，而要在其派生出的非抽象类中实现。

注：抽象方法的类必须为抽象类,不能在类中实现; 在派生类实现中, 使用 REDEFINTION 对类方法重新定义，重新定义过程中不可使用 SUPER。

2．最终类和最终方法

语法：

```
CLASS <class> DEFINITION FINAL.
  METHODS meth FINAL...
ENDCLASS.
```

最终类（Final Class）是不能被继承的类，最终方法是不可重新定义的方法，不一定只出现在最终类中，但最终类中的所有方法都是最终的，而最终类中无需指明 FINAL。

注：类的种别决定该类的实例可以被所有用户创建，只能被其本身或派生类创建或只能通过其自身方法创建，ABAP 常用类型如下:

- ◆ Final：全局类不能在程序中被继承，即最终类。
- ◆ Abstract：只能被继承不能被实例化。
- ◆ For Testing：ABAP 单元的测试类。

最终类可同时为抽象类，但只能包含静态成员；最终方法则不可以同时为抽象方法。

1.1.2　多态

相同名称的方法在不同的类中呈现出不同的行为，当一个相同的方法在不同的类中有不同的实现就是多态。在 ABAP 中，多态可以通过继承和接口实现方法。

多态性是指相同的用户操作在不同类中有不同的实现形式，在继承中，多态性是通过方法重载及引用变量实现的，即子类可重新定义并以不同的方式实现基类中的公有或保护方法，此时，基类中的方法需为实例方法。派生类声明部分需进行 METHOD meth REDEFINTION，该方法必须和基类中的方法具有相同接口，但可通过不同的代码实现。

当访问基类时，将采取基类的原始方法实现；而访问派生类时，则采用新的派生类方法实现，原有基类方法将被屏蔽，包括通过自身引用 me-> meth，不过可使用 SUPER 来指定其基类 CALL METHOD super->meth。

注：只有在方法重载实现代码内部才能使用 SUPER。最终方法不能在子类中重定义，最终类没有子类。

1.1.3　继承

继承可以定义多个类之间的执行关系，可以从已存在类的基础上建立新的派生类，派生类可以继承原有类的方法与属性，或者添加新的类成员；高级类的运行方式也可能得到调整和扩展。

继承允许从一个类中引出新的类。新的类叫子类（Sub Class），被引出的类叫父类（Super Class）。

1．继承的声明

语法：

```
CLASS <class> DEFINITION INHERITING FROM <superclass>.
  ……（省略完整代码）
ENDCLASS.
```

在子类中，父类的类型为 Public 和 Protected 的组件才是可见的，并且继承只能是单继承。所有 ABAP 对象的根类是 OBJECT。

2．方法的重定义

语法：

```
CLASS <class> DEFINITION INHERITING FROM <superclass>.
  METHODS meth REDEFINITION...
ENDCLASS.
```

方法的重用使用声明 REDEFINITION 在 METHODS 中定义，但是不能改变它的接口。用 ME->代替自身的对象变量，用 SUPER->代替父类的对象变量。

在类定义时，使用 INHERTING FROM 附加项可以指定各个子类之间的继承关系。ABAP 中继承为单一继承，即一对多。

3．继承过程中各个成员的可见性

- 派生类中的公有成员以及基类的公有成员，都可以通过"->"在类外部获得。
- 派生类的被保护成员以及基类的被保护成员，都不可以通过"->"在类外部获得，但可在派生类内部使用。
- 派生类中的私有成员只包括其本身私有部分定义的成员，只能在派生类内部使用。

继承类和基类的公有成员和被保护成员享有共同的命名空间，而私有成员则可在不同类之间出现重名情况。基类应用变量可以指向其继承类对象，反之则会被视为错误的类型转换。

1.1.4 面向对象的成员

在类中能够使用 TYPE 语句来定义自己的 ABAP 数据类型，类型不是实例相关的，对类中的所有对象只存在一次。

1．成员类型

ABAP 中可以定义三种不同的成员类型，分别是属性、方法和事件，各成员的可见性以及生存周期在类定义的同时被确定。

（1）属性

属性是在类内部的数据对象，用于描述类的状态。属性可以由多个数据类型组成，对象的说明由属性内容定义。一种类型的属性是关联变量，它允许建立和定位对象。关联变量能够在类内部定义，也允许从类中访问对象。

实例属性

实例属性的内容定义了专用实例，可以用 DATA 语句声明。

静态属性

静态属性的内容定义了类的声明，对类的所有实例有效。静态属性对每个类存在一次，可以用 CLASS-DATA 语句来声明它们，它们在类的全部运行时间都可以被访问。

常量

常量是特殊的静态属性，在声明时设置值，然后就不能再修改。声明使用 CONSTANTS 语句。

（2）方法

方法是对象行为的实现部分，可以将其视为是类中的过程，用于访问或修改对象状态。

方法的定义和参数接口与函数模块相同，可以在声明部定义。

```
语法：  METHODS <meth> IMPORTING.. [VALUE（]<ii>[）] TYPE type [OPTIONAL]..
                              EXPORTING.. [VALUE（]<ei>[）] TYPE type [OPTIONAL]..
                              CHANGING.. [VALUE（]<ci>[）] TYPE type [OPTIONAL]..
                              RETURNING VALUE（<r>）
                              EXCEPTIONS.. <ei>..
```

然后在执行部实现，语法如下。

```
语法：  METHOD <meth>.
        ...
        ENDMETHOD.
```

本地数据类型和方法的定义与其他 ABAP 程序相同，可以用语句 CALL METHOD 来调用。

```
语法：  CALL METHOD <meth> EXPORTING... <ii> =.<f i>...
                           IMPORTING... <ei> =.<g i>...
                           CHANGING ... <ci> =.<f i>...
                           RECEIVING          r = h
                           EXCEPTIONS... <ei> = rc i...
```

参数解释：

- Importing：指定一个或多个输入参数。
- Exporting：指定一个或多个输出参数。
- Changing：指定一个或多个输入/输出参数。
- Value：指定参数传递形式，分为值传递和引用传递两种形式。默认为引用传递，如果使用 Value 选项则为值传递。
- Type：指定参数类型，每一个参数都必须有指定的类型。
- Optional：指定该参数为可选时，程序将使用该参数类型的初始值进行传递，或者是使用 Default 选项后指定的默认值。
- Returning：该选项可以替代 Importing 和 Exporting。

实例方法

可以用 METHODS 语句声明实例方法，它们可以访问类中的所有属性，能够触发所有的事件。

静态方法

可以使用 CLASS-METHODS 语句声明静态方法，它们只能够访问静态属性和触发静态事件。

构造方法（类构建器）

同普通方法一样，可以通过 CALL METHOD 调用，当建立一个对象（CONSTRUCTOR）或第一次访问类的部件时（CLASS_CONSTRUCTOR），有两个叫 CONSTRUCTOR 和 CLASS_CONSTRUCTOR 的专用方法能够自动被调用。

（1）构造方法

- 构造方法是运行时环境自动调用的一种方法，用于为对象设定一个初始化状态，不能

在程序运行过程中由程序代码调用。与其他方法类似，构造方法也存在静态和实例两种形式。

- 构造方法不一定需要在类中定义，系统通常会自动生成一个默认的构造方法，将对象状态设置为初始值。
- 构造方法中不包含任何形式的输出参数，其功能只是定义对象状态，而不是对其进行改变。
- 构造方法在 Create Object 语句中被调用，根据实例构造方法的定义，该语句本身也可以包含输出参数 Exporting 和异常返回值 Exceptions 选项。
- 类构造方法属于静态方法，只能在程序中被调用一次，即第一次使用该类时被调用，该方法也必须在公有部分声明，其名称必须为 Class_Constructor。
- 类构造方法不包含接口，而且在其中只能访问类的静态属性。

（2）析构方法

在某些面向对象语言中还存在析构方法的概念，该方法在对象删除过程中被调用，目前 ABAP 对象中尚未提供该方法的实现形式。

每个类都有一个隐含的方法，在 Java 或 C++中叫构造函数；在一般语言中，构造函数和类名相同，但是在 ABAP 中，有一个保留的名字，叫做 Constructor。当调用语句 Create Object 创建对象的时候，这个函数会被执行。

构造函数 Constructor 可以分为实例 Constructor 以及静态 Constructor。对于实例 Constructor（声明的时候使用 Methods），可以有 Import 参数以及 Exceptions，而静态 Constructor（声明的时候使用 Class-Methods），没有参数传递。

注意：静态构造函数一定会在以下动作之前执行。

- 创建对象，Create Object。
- 访问静态属性，Class=>Attribute。
- 调用静态方法，Call Method_Class=>Method。
- 注册一个静态事件句柄。

（3）事件

用于一个类对象发布其状态的改变，因而其他对象可以捕获该方法并做出响应。方法是类的内部处理过程，定义了一个对象的行为。它可以访问所有类的属性，也可以修改对象的数据内容，还可以有一个参数接口，用户可以通过带参数调用，接收返回值。私有属性只能通过方法类修改。

对象或者类能够在其他对象或类中通过事件来触发事件处理的方法。在一个一般的方法调用中，一个方法可以被许多用户调用。当一个事件被触发，一系列事件处理方法可以被调用。在运行之前，触发器和处理方法之间的联系没有被建立。在一般的方法调用中，调用的 Program 决定了它希望调用的方法，这些方法必须存在。根据事件，处理方法确定它希望响应的事件。对所有的事件不需要注册处理方法。

类的事件可以在类的方法中通过使用 RAISE EVENT 语句触发。可以通过使用字句 FOR EVENT <evt> OF <class>声明同一个或不同的类的方法作为一个事件处理的方法。

事件有一个与方法相似的参数接口，但是只有输出参数时这些参数通过触发器（RAISE

EVENT 语句）传递给事件处理方法，它把它们作为输入参数接收。

触发器和处理方法之间的联系通过 SET HANDLER 语句动态建立。触发器和处理方法可以是对象或类，这取决于使用的是实例还是静态事件或静态处理方法。当一个事件被触发，相关事件处理方法在所有注册的处理类中执行。

实例事件

可以用 EVENTS 语句声明实例事件。一个实例事件只能在实例方法中被触发。

静态事件

可以用 CLASS-EVENTS 语句声明静态事件，所有的方法（静态或实例）都可以触发静态事件。静态事件是只能在静态方法中被触发的事件类型。

1）触发器和处理事件

在 ABAP 对象中，触发器和处理一个事件意味着特定的方法作为触发器和触发事件执行。对于其他方法的相应处理器，这意味着当事件触发时处理方法被执行。

在 ABAP 对象中处理事件有以下两种方法。

■ 在声明部声明事件。

■ 使用它的一个方法触发事件。

事件声明

声明实例事件语法如下所示。

语法： EVENTS <evt> EXPORTING... VALUE(<e_i>) TYPE type [OPTIONAL]..

声明静态事件语法如下所示。

语法： CLASS-EVENTS <evt>...

两个语句都有同样的语法，EXPORTING 字句用于指定传递给事件处理的参数。参数一般是值传递。实例事件一般包含默认参数 SENDER，它具有与事件声明的接口或类型相关的类型。

触发事件

实例事件可以被类中的所有方法触发，静态事件可以被静态方法触发。

语法：RAISE EVENT <evt> EXPORTING... <e_i> = <f_i>...

对每个没有被定义为正式参数的<e_i>，必须在EXPORTING字句中传递一个相应的实际参数<f_i>。自定义的ME自动被传递给默认参数SENDER。

2）事件处理

事件处理使用指定的方法，处理事件方法的要求有以下两个。

■ 被定义成为事件的事件处理方法。

■ 在执行时为事件注册。

事件处理方法的声明

所有的类可以包含针对其他类事件的处理方法。当然，也可以在同一个类中定义处理方法。实例方法的声明语句如下。

语法： METHODS <meth> FOR EVENT <evt> OF <cif> IMPORTING.. <e_i>..

静态方法使用 CLASS-METHODS 来代替 METHODS。<evt>是在类或接口中的一个事件声明。

一个处理方法的接口只能包含在事件<evt>声明中定义的正式参数中。参数属性也由事件使用。处理方法不一定需要使用 RAISE EVENT 语句所传递的所有参数，如果希望使用默认参数 SENDER，必须在接口中包含。这个参数允许一个实例事件处理访问触发器，例如，允许它返回结果。

一旦在类中定义了事件处理方法，这意味着类的实例或类本身在理论上能够处理在一个方法中触发的事件<evt>。

注册事件处理方法

为一个事件处理方法响应事件，必须在运行触发器时确定哪个用于响应。通过以下语句实现。

语法： SET HANDLER... <h_i>... [FOR]...

它将处理方法清单和相应的触发方法联系起来。事件有四个不同的方法：

- 在类中声明的实例事件。
- 在接口中声明的实例事件。
- 在类中声明的静态事件。
- 在接口中声明的静态事件。

针对以上四种情况，SET HANDLER 语句语法和作用是不同的。

对于实例事件，可以使用 FOR 字句来指定实例需要注册的处理。也可以指定一单个实例作为触发器，使用关联变量<ref>。

语法： SET HANDLER... <h_i>...FOR <ref>.

或者可以为所有能够触发事件的实例注册处理方法。

语法： SET HANDLER... <h_i>...FOR ALL INSTANCES.

然后，注册器为在注册处理时还没有建立的触发实例提供事件。可以使用 FOR 字句定义静态事件。

语法： SET HANDLER... <h_i>...

注册器自动提供给整个类，或者给所有实现含静态事件接口的类。对于接口，注册器还应用于那些在动态运行中直到处理程序的类。

3）事件处理的生命期

在 RAISE EVENT 后，所有注册的处理方法在下一个语句处理之前被执行（同步事件处理）。如果一个处理方法自己触发事件，则它的处理方法在初始处理方法之前。为避免无穷的递归，事件处理一般只嵌套 64 层。

处理方法根据它们被定义的顺序执行，一旦事件处理被动态注册，则不能认为它们在特定的顺序执行，相反，必须认为所有的事件处理会同步执行。

注：在 ABAP 中，接口需要另外实施（其实就是在类中实施），同时又独立于类。在接口

中只有声明部分，而没有具体的实施部分。

- ■ 接口必须在类的公共部分列出（Public Section）。
- ■ 接口中的操作在类中实现，并且作为类的方法；所有接口中的方法必须在类的实现部分实施。
- ■ 在接口中定义的属性、事件、常量、类型等在声明它的类中自动可见。
- ■ 实施接口的方法：Interface～Component Name。

2．成员的可见性

成员的可见性是基于封装原则确定的。

（1）PUBLIC SECTION

公有部分定义的类成员可以被所有对象使用，包括类方法及其派生类中定义的方法。这些组件构成了类的外界接口部分。

（2）PROTECTED SECTION

保护部分中定义的类成员只能被类及其派生类中的方法使用，外部不可见，这些组件连同公有组件构成了类和派生类对象之间的接口。在 4.5B 中，继承是无效的，所以 Protected 范围与 Private 相同。

（3）PRIVATE SECTION

类成员只能被类本身的方法使用。

注：

公有属性（Public Attributes）定义在 Public Section 中，其内容可以在类的外部看到或者修改。尽量减少公共属性，其原因可以参考面向对象语言的特性——封装。尽可能在类的内部修改属性，留给外部的只有接口，避免外部直接操作类里面的属性（或称数据）。

只能在类的内部访问或者修改的数据，定义在 Private Section 里面。

在类中，每个静态属性都是唯一的。属性的值在所有类的实例中是相同的。用 class-data 声明。

保护属性（Protected Attributes）定义在 Protected Section 里面，其作用是更改区域里面的属性或者方法，不能被外部调用，但是在它的子类中是可见的。也就是说，其子类可以访问 Protected Section 中的数据，这和 Private Section 是不同的。

Public Method、Private Method、Static Method、Protected Method 及其对应的属性同上。

1）使用对象的一般步骤

- ● 定义类的声明与方法实现。
- ● 使用 DATA 语句中的 TYPE REF TO 选项参照类类型声明引用变量。
- ● 使用 CREAT OBJECT 语句创建对象。
- ● 通过-> 或=>运算符访问对象或类组件。

2）访问对象组件

对象是 ABAP 程序所获得内存空间里的一个类的实例。首先在 DATA 声明里用 TYPE REF TO <类>声明 ABAP 数据类型，然后用 CREATE OBJECT 创建类的实例。也可以用 MOVE 对兼容的对象之间赋值，以后就可以使用。

3）访问内容语法格式

- 一个对象的实例属性或静态属性 oref-> attr。
- 类的静态属性 class=>attr。
- 在类内部访问自身实例属性或静态属性 me->attr 或 attr。
- 对象的实例属性或静态方法 CALL METHOD oref->meth。
- 类的静态方法 CALL METHOD class=>meth。
- 在类内部访问自身实例方法或静态方法 CALL METHOD me->attr 或 CALL METHOD attr。

1.1.5 接口

当有些类具有相同的功能却有不同的行为时，就产生了接口的概念。接口这样描述自己："对于实现所有类，看起来都应该像现在这个样子"。因此，采用了一个特定接口，所有代码都知道对于那个接口可能会调用什么方法。所以常把接口用于建立类和类之间的一个"协议"。有些面向对象的程序设计语言采用了一个名为"Protocol"（协议）的关键字，它做的便是与接口相同的事情。接口只有声明部分，并不提供实现的细节。

1．接口的定义与实现

对象引用包括类引用和接口引用，通过 ABAP 类的实例及引用可构建各种商业应用模块及其中的元素。

接口是一个独立结构，可以在其中定义一些成员并在具体类中实现，其作用是对类中已定义的成员进行扩展。实现接口后，其成员将成为公有成员，但类可自行对接口中的方法以自身特定形式实现。

1）定义接口：与类的定义相似，接口作为独立 R/3 Repository 的对象在 Class Builder 中定义，也可以依附于程序在 ABAP 程序中定义（本地类）。

2）实现接口：接口没有自己的实例，因而不需要进行实现，其实现通过具体类进行。

注：接口 intf 中的成员 icomp 在类内部实现过程以名称 inf～imeth 的形式出现。

接口的实现只能出现在公有部分，接口中定义的所有组件都将被添加为该类的公有成员。在类的实现部分须包含所有的接口方法。

一个接口可被任意多个不同类实现，该接口中定义的成员集在各个类中名称相同，然而实现方法不同，也就体现了多态性。若一个类中除了接口之外没有定义任何类自身的公有成员方法，则接口就成为了该类的全部"外部接口"。

2．接口的定义方法

接口跟类一样，也有全局接口和本地接口。

1）全局接口的定义由类定义工具实现（事务码 SE24）。

2）本地接口的定义

本地接口封装在 INTERFACE...ENDINTERFACE 中，包含所有的部件（属性、方法、事件）。所有的部件不能包含可视部分，并且自动会被声明为 PUBLIC。

```
INTERFACE <intf>.
```

```
        ……（省略完整代码）
ENDINTERFACE
```

由于接口没有实现部分，它是在类中实现的。类中对接口的声明如下：

```
INTERFACES <intf>.
```

3）类中接口部件的实现

类实现部分对接口方法的实现如下：

```
METHOD <intf～imeth>.
    ……（省略完整代码）
ENDMETHOD.
```

3．接口引用

引用变量也可以基于接口声明变量，即接口引用（一种对象引用）。除通过类引用访问之外，还可以通过引用进行访问，因为接口引用可以指向任何实现了该接口的对象。

通过 TYPES 或 DATA 语句中的 TYPE REF TO 项进行：

```
TYPES|DATA iref TYPE REF TO intf.
```

其中 intf 是全局或程序中已定义的接口，iref 接口引用 iref->icomp 访问。其可访问当前指向对象的接口中的可见成员，其方法可以在对象类中实现。

1）通过接口引用访问对象

访问一个类对象，须先声明一个基于该类的引用变量，若该类为一个接口实现，则可将该引用变量赋给一个接口变量，接口变量名 iref，类引用 cref，iref=cref。若接口中还含有实例属性 attr 和实例方法 meth，如：

```
cref->inth～attr.       CALLMETHOD cref->intf～meth.
iref->attr              CALLMETHOD iref->meth.
```

对于接口中定义静态成员，若是常量，则只能通过接口引用访问；其他静态接口成员，可通过类本身或类引用。如：

```
intf=>const
class=>intf～attr     CALL METHOD class=>intf～meth.
```

2）在接口引用间赋值

可将接口引用分配给不同引用变量，还可以在类引用和接口引用之间相互赋值。若类引用 cref 为相同的接口或类，可直接赋值。如：

```
iref1=iref2. iref=cref.
```

两者不参照同一接口声明，或 iref2 是参照接口 iref1 的一个成员，强制赋值运算符 ？= 可使接口数据间进行数据交换。

```
MOVE iref ？TO cref.
cref ？= iref.          //不允许时会触发 MOVE_CAST_ERROR。
```

类引用为 Object 参照或强制赋值；接口引用与类引用 cref 的参照类相同。

3）接口的使用

先用 TYPE REF TO 和 DATA 声明接口变量，但不能用 CREAT 来创建实例，而是通过参考其他已存在的类变量或接口变量来代替。

（1）接口实例组件的两种使用方法

通过类实例使用：

```
使用属性 <attr>：  <cref>-><intf～attr>
调用方法 <meth>：  CALL METHOD <cref>-><intf～meth>
```

通过接口变量使用：

```
使用属性<attr>：  < iref>-><attr>
调用方法<meth>：  CALL METHOD <iref>-><meth>
```

（2）接口静态组件的使用方法

常量可以使用：

```
使用常量 <const>：  < intf>=><const>
```

其他静态组件必须使用：

```
使用静态属性 <attr>：  < class>=><intf～attr>
调用静态方法 <meth>：  CALL METHOD <class>=><intf～meth>
```

1.2 本地类的实现

1.2.1 程序设计

程序源码设计如下：

```
*----------------------------------------------------------------------*
*
* Program ID              ： ZYOO_TEST
* Program Description     ： 成绩表
* Create   User           ： BCUSER
* Create   Date           ： 2011/02/20
*
*----------------------------------------------------------------------*
* Function Overview
*  以面向对象方式把成绩表显示出来
*  技术点：本地面向对象定义，声明，使用
*----------------------------------------------------------------------*
*
* Change List            ：
*      change NO.   Change Date     Change User     Change Detail
```

```
*     0000001     2011/02/20     BCUSER
*----------------------------------------------------------------------*
REPORT ZYOO_TEST
   NO STANDARD PAGE HEADING
   LINE-SIZE 120
   LINE-COUNT 90(03)
   MESSAGE-ID ZTESTMSG.

*----------------------------------------------------------------------*
* INCLUDE
*----------------------------------------------------------------------*
INCLUDE：  ZTEST0N,    "常量定义
           ZTEST0F.    "Form 定义

*----------------------------------------------------------------------*
* TABLES：
*----------------------------------------------------------------------*
TABLES：  ZTTEST_T.

*----------------------------------------------------------------------*
* CONSTANTS
*----------------------------------------------------------------------*
CONSTANTS：  LINE88 TYPE I VALUE 88.

*----------------------------------------------------------------------*
* DATA
*----------------------------------------------------------------------*
DATA：  BEGIN OF FLG,
        CLS_FRST TYPE C VALUE CNS-OFF,            "班级起始
        END OF FLG.

*----------------------------------------------------------------------*
* CLASS-DATA
*----------------------------------------------------------------------*
*  接口定义
INTERFACE TEST .
   METHODS：  GET_NAME ,GET_DETAIL,GET_ALL.
   EVENTS   NOT_FOUND EXPORTING VALUE(ID)   TYPE ANY.
ENDINTERFACE.                     "TEST

*----------------------------------------------------------------------*
*          CLASS CLS_TEACHER DEFINITION
*----------------------------------------------------------------------*
*          教师信息表
*----------------------------------------------------------------------*
CLASS CLS_TEACHER DEFINITION .
```

```
  PUBLIC SECTION.
    DATA ZTNUMBER TYPE ZMTNUMBER_T-ZTNUMBER. "教师番号
*   接口使用
    INTERFACES： TEST.
    ALIASES ：GET_TEACHER_NAME    FOR TEST～GET_NAME ,
              GET_TEACHER_DETAIL  FOR TEST～GET_DETAIL ,
              GET_TEACHER_ALL     FOR TEST～GET_ALL ,
              NO_TEACHER_FOUND    FOR TEST～NOT_FOUND.
*   事件触发方法
    METHODS ：ERROR_HANDLE        FOR EVENT NO_TEACHER_FOUND
    OF CLS_TEACHER IMPORTING ID.

  PROTECTED SECTION.
    DATA： ZTNAME   TYPE ZMTNUMBER_T-ZTNAME,  "姓名
           ZTSEX    TYPE ZMTNUMBER_T-ZTSEX,   "性别
           ZZMOBILE TYPE ZMTNUMBER_T-ZZMOBILE."联系电话
  PRIVATE SECTION.
    DATA： TEACHER TYPE   ZMTNUMBER_T .
ENDCLASS.                   "CLS_TEACHER DEFINITION

*---------------------------------------------------------------------*
*         CLASS CLS_TEACHER IMPLEMENTATION
*---------------------------------------------------------------------*
*         教师信息表
*---------------------------------------------------------------------*
CLASS CLS_TEACHER IMPLEMENTATION.
* 取得教师姓名
  METHOD GET_TEACHER_NAME   .
    SELECT   SINGLE ZTNAME                "姓名
           FROM   ZMTNUMBER_T             "教师信息表
           INTO ME->ZTNAME                "姓名
            WHERE   ZTNUMBER   = ME->ZTNUMBER .
    IF   SY-SUBRC <> 0.
      RAISE EVENT NO_TEACHER_FOUND EXPORTING ID = ME->ZTNUMBER.
    ENDIF.

  ENDMETHOD.                              "GET_TEACHER_NAME

* 取得教师详细
  METHOD GET_TEACHER_DETAIL   .
    SELECT   SINGLE ZTNAME ZTSEX ZZMOBILE
            FROM   ZMTNUMBER_T
            INTO (ME->ZTNAME,ME->ZTSEX ,ME->ZZMOBILE )
             WHERE   ZTNUMBER   = ME->ZTNUMBER .
    IF   SY-SUBRC <> 0.
      RAISE EVENT NO_TEACHER_FOUND EXPORTING ID = ME->ZTNUMBER.
```

```
    ENDIF.
  ENDMETHOD.                         "GET_TEACHER_DETAIL

*  取得教师信息
  METHOD GET_TEACHER_ALL    .
    SELECT    SINGLE *
              FROM    ZMTNUMBER_T
              INTO ME->TEACHER
               WHERE    ZTNUMBER    = ME->ZTNUMBER .
    IF    SY-SUBRC <> 0.
      RAISE EVENT NO_TEACHER_FOUND EXPORTING ID = ME->ZTNUMBER.

    ENDIF.
  ENDMETHOD.                         "GET_TEACHER_ALL
*  输出错误信息
  METHOD ERROR_HANDLE .
    MESSAGE E001 WITH ID.
  ENDMETHOD.                         "error_handle

ENDCLASS.                        "CLS_TEACHER IMPLEMENTATION
*---------------------------------------------------------------------*
*          CLASS CLS_TEACHER DEFINITION
*---------------------------------------------------------------------*
*          学生信息表
*---------------------------------------------------------------------*
CLASS CLS_STUDENT DEFINITION CREATE PROTECTED FRIENDS TEST .
  PUBLIC SECTION.
    DATA ZZNUMBER TYPE ZMNUMBER_T-ZZNUMBER.                "学生编号
*   构造函数，初始化该类的实例
    METHODS：CONSTRUCTOR IMPORTING SID TYPE ZMNUMBER_T-ZZNUMBER.
*   接口使用
    INTERFACES：  TEST.
    ALIASES  ：GET_STUDENT_NAME     FOR TEST～GET_NAME ,
             GET_STUDENT_DETAIL FOR TEST～GET_DETAIL ,
             GET_STUDENT_ALL       FOR TEST～GET_ALL ,
    METHODS：ERROR_HANDLE FOR EVENT NO_STUDENT_FOUND OF CLS_STUDENT
    IMPORTING ID.

  PROTECTED SECTION.
    DATA：   ZZNAME      TYPE ZMNUMBER_T-ZZNAME,        "姓名
          ZZSEX             TYPE ZMNUMBER_T-ZZSEX,          "性别
          ZZFNAME        TYPE ZMNUMBER_T-ZZFNAME,    "父亲
          ZZMNAME        TYPE ZMNUMBER_T-ZZMNAME,    "母亲
          ZZMOBILE TYPE ZMNUMBER_T-ZZMOBILE,   "联系电话
          ZZADDR       TYPE ZMNUMBER_T-ZZADDR.      "家庭住址
  PRIVATE SECTION.
```

```
    DATA：  STUDENT TYPE     ZMNUMBER_T .
ENDCLASS.                    "CLS_STUDENT DEFINITION

*------------------------------------------------------------------*
*         CLASS CLS_STUDENT IMPLEMENTATION
*------------------------------------------------------------------*
*         学生信息表
*------------------------------------------------------------------*
CLASS CLS_STUDENT IMPLEMENTATION.
* 构造函数声明
  METHOD CONSTRUCTOR.
    ME->ZZNUMBER = SID.
    SET HANDLER ERROR_HANDLE FOR ALL INSTANCES.
  ENDMETHOD.                    "CONSTRUCTOR
* 取得学生姓名
  METHOD GET_STUDENT_NAME   .
    SELECT  SINGLE ZZNAME
          FROM  ZMNUMBER_T
          INTO ME->ZZNAME
          WHERE  ZZNUMBER  = ME->ZZNUMBER .
    IF  SY-SUBRC <> 0.
      RAISE EVENT NO_STUDENT_FOUND EXPORTING ID = ME->ZZNUMBER.
    ENDIF.

  ENDMETHOD.                    "GET_STUDENT_NAME
* 取得学生详细
  METHOD GET_STUDENT_DETAIL   .
    SELECT  SINGLE ZZNAME ZZSEX ZZFNAME ZZMNAME ZZMOBILE ZZADDR
           FROM  ZMNUMBER_T
           INTO （ME->ZZNAME,ME->ZZSEX ,ME->ZZFNAME, ME->ZZMNAME,
                 ME->ZZMOBILE ,ME->ZZADDR）
            WHERE  ZZNUMBER  = ME->ZZNUMBER .
    IF  SY-SUBRC <> 0.
      RAISE EVENT NO_STUDENT_FOUND EXPORTING ID = ME->ZZNUMBER.
    ENDIF.
  ENDMETHOD.                    "GET_STUDENT_DETAIL
* 取得学生信息
  METHOD GET_STUDENT_ALL   .
    SELECT  SINGLE *
           FROM  ZMNUMBER_T
           INTO ME->STUDENT
            WHERE  ZZNUMBER  = ME->ZZNUMBER .
    IF  SY-SUBRC <> 0.
      RAISE EVENT NO_STUDENT_FOUND EXPORTING ID = ME->ZZNUMBER.
    ENDIF.
  ENDMETHOD.                    "GET_STUDENT_ALL
```

```
  METHOD ERROR_HANDLE .
    MESSAGE E001 WITH ID.
  ENDMETHOD.                    "error_handle

ENDCLASS.                    "CLS_STUDENT IMPLEMENTATION
*----------------------------------------------------------------------*
*           CLASS CLS_CLASS    DEFINITION
*----------------------------------------------------------------------*
*           班主任表
*----------------------------------------------------------------------*
CLASS CLS_CLASS DEFINITION INHERITING FROM CLS_TEACHER .
  PUBLIC SECTION.
    DATA：  ZZCLASS     TYPE ZCCLASS_T-ZZCLASS ,          "班级
            ZZCLSNAME   TYPE STRING,                      "班级名称
            ZZTEANOM    TYPE ZCCLASS_T-ZZTEANOM,          "班主任
            ZZTEANOS    TYPE ZCCLASS_T-ZZTEANOS,          "副班主任
            ZTNAMEM     TYPE ZMTNUMBER_T-ZTNAME,          "班主任名
            ZTNAMES     TYPE ZMTNUMBER_T-ZTNAME.          "副班主任名
*   取得班级名称
    METHODS：GET_CLASS_DOMAIN .
    EVENTS：  NO_CLASS_FOUND
      EXPORTING VALUE（CLSID）  TYPE ZCCLASS_T-ZZCLASS.
*   声明友类
    DATA TEACHER TYPE REF TO CLS_TEACHER.
*   取得班级详细信息
    METHODS：  GET_CLASS.
*   取得班级错误处理
    METHODS：NODATA_HANDLE FOR EVENT NO_CLASS_FOUND OF CLS_CLASS
    IMPORTING CLSID.

ENDCLASS.                    "XXXX DEFINITION

*----------------------------------------------------------------------*
*           CLASS CLS_CLASS IMPLEMENTATION
*----------------------------------------------------------------------*
*           班主任表
*----------------------------------------------------------------------*
CLASS CLS_CLASS IMPLEMENTATION.
* 班级信息取得
  METHOD GET_CLASS    .

    SELECT   SINGLE ZZTEANOM ZZTEANOS
       FROM   ZCCLASS_T
       INTO  （ME->ZZTEANOM,ME->ZZTEANOS）
            WHERE   ZZCLASS     = ME->ZZCLASS ."班级
    IF   SY-SUBRC <> 0.
```

```
      RAISE EVENT NO_CLASS_FOUND EXPORTING CLSID = ME->ZZCLASS.
    ELSE.
      SET HANDLER ERROR_HANDLE FOR ALL INSTANCES.
      ME->ZTNUMBER = ZZTEANOM.
      CALL METHOD ME->GET_TEACHER_NAME.
      ME->ZTNAMEM = ME->ZTNAME .
      ME->ZTNUMBER = ZZTEANOS.
      CALL METHOD ME->GET_TEACHER_NAME.
      ME->ZTNAMES = ME->ZTNAME .
    ENDIF.

  ENDMETHOD.                    "GET_CLASS
* 取得班级名称
  METHOD GET_CLASS_DOMAIN    .

    SELECT    SINGLE    DD07T～DDTEXT                "域固定值文本
    FROM    DD07T INNER JOIN    DD07L                "内结合
    ON    DD07T～DOMNAME = DD07L～DOMNAME            "结合条件
    AND DD07T～VALPOS    = DD07L～VALPOS
    WHERE DD07L～DOMNAME = 'ZZCLASS_D'               "域名相等
    AND DD07L～DOMVALUE_L = ME->ZZCLASS.             "域固定值相等
    IF    SY-SUBRC <> 0.
      RAISE EVENT NO_CLASS_FOUND EXPORTING CLSID = ME->ZZCLASS.
    ENDIF.
  ENDMETHOD.                    "GET_CLASS_DOMAIN

  METHOD NODATA_HANDLE .
    MESSAGE E001 WITH CLSID.
  ENDMETHOD.                    "error_handle

ENDCLASS.                    "CLS_CLASS IMPLEMENTATION
*---------------------------------------------------------------------*
*           CLASS CLS_TEST DEFINITION
*---------------------------------------------------------------------*
*           成绩表
*---------------------------------------------------------------------*
CLASS CLS_TEST DEFINITION INHERITING FROM CLS_CLASS    .
  PUBLIC SECTION.

    DATA：    ZZYEAR TYPE ZZTEST_S-ZZYEAR,             "学年
          ZZSTDNOF TYPE ZZTEST_S-ZZSTDNO,            "学号 From
          ZZSTDNOT TYPE ZZTEST_S-ZZSTDNO,            "学号 To
          TEST TYPE TABLE OF ZZTEST_S ,              "成绩表
          H_TEST TYPE ZZTEST_S .                     "成绩表

    EVENTS：    NO_TEST_DATA
```

```
    EXPORTING VALUE（CLASS）        TYPE ZCCLASS_T-ZZCLASS
              VALUE（YEAR）         TYPE ZTTEST_T-ZZYEAR.
  METHODS：  GET_TEST IMPORTING  CLASS TYPE ZCCLASS_T-ZZCLASS
                                 YEAR TYPE ZTTEST_T-ZZYEAR.
  METHODS：NOTEST_HANDLE FOR EVENT NO_TEST_DATA OF CLS_TEST
  IMPORTING CLASS YEAR.

ENDCLASS.                        "CLS_TEST DEFINITION

*---------------------------------------------------------------------*
*       CLASS CLS_TEST IMPLEMENTATION
*---------------------------------------------------------------------*
*       成绩表
*---------------------------------------------------------------------*
CLASS CLS_TEST IMPLEMENTATION.
* 成绩表数据抽取
 METHOD GET_TEST   .
  SELECT       ZZCLASS              "班级
              ZZYEAR                "学年
              ZZSTDNO               "学号
              ZZTENKOK              "语文
              ZZTENSAN              "数学
              ZZTENRIK              "自然
              ZZTENSHA              "思品
    FROM ZTTEST_T                   "成绩表
    INTO CORRESPONDING FIELDS OF TABLE TEST          "成绩内部表
    WHERE ZZCLASS   =   CLASS       "班级
    AND    ZZYEAR = YEAR            "学年
  IF   SY-SUBRC <> 0.
    RAISE EVENT NO_TEST_DATA
      EXPORTING CLASS = CLASS YEAR = YEAR.
  ELSE.
    DATA STUDENT   TYPE REF TO CLS_STUDENT .
    LOOP AT TEST INTO H_TEST.
      CREATE OBJECT STUDENT
        EXPORTING
          SID = H_TEST-ZZSTDNO.

      CALL METHOD STUDENT->GET_STUDENT_NAME.
      H_TEST-ZZNAME = STUDENT->ZZNAME.
      MODIFY TEST FROM H_TEST.
    ENDLOOP.
  ENDIF.
 ENDMETHOD.                        "GET_TEST

 METHOD NOTEST_HANDLE.
```

```
    MESSAGE E005 WITH CLASS YEAR.
  ENDMETHOD.                          "error_handle

ENDCLASS.                       "cls_TEACHER IMPLEMENTATION
*  成绩表对象声明
DATA TEST TYPE REF TO CLS_TEST.
*---------------------------------------------------------------------*
* PARAMETERS
*---------------------------------------------------------------------*
PARAMETERS： P_CLASS       TYPE ZTTEST_T-ZZCLASS            "班级
                           OBLIGATORY .                     "必须

PARAMETERS P_YEAR TYPE ZTTEST_T-ZZYEAR OBLIGATORY.     "学年

*---------------------------------------------------------------------*
* SELECT-OPTIONS
*---------------------------------------------------------------------*
SELECT-OPTIONS S_STDNO     FOR ZTTEST_T-ZZSTDNO             "学号
                           MATCHCODE OBJECT ZZMCSBID.

*---------------------------------------------------------------------*
* INITIALIZATION
*---------------------------------------------------------------------*
INITIALIZATION.

*  数据初始化
  PERFORM DATA_INIT
              USING
                P_CLASS      "班级
                P_YEAR.      "学年
  CREATE OBJECT TEST.
  SET HANDLER TEST->NOTEST_HANDLE FOR ALL INSTANCES.
  SET HANDLER TEST->NODATA_HANDLE FOR ALL INSTANCES.

*---------------------------------------------------------------------*
* AT SELECTION SCREEN
*---------------------------------------------------------------------*
AT SELECTION-SCREEN.

*  选择画面项目检查-班级
  PERFORM CLASSOO_INPUT_CHECK.

*  选择画面项目检查-学年
  PERFORM YEAR_INPUT_CHECK USING P_YEAR TEXT-001.

*---------------------------------------------------------------------*
```

```
*START-OF-SELECTION
*---------------------------------------------------------------------*
START-OF-SELECTION.

*  报表数据初始化
   PERFORM REPORT_DATA_INIT.
*  报表数据选择
   PERFORM REPORT_DATA_SELECT.
*  报表数据打印
   PERFORM REPORT_DATA_EXE.

*---------------------------------------------------------------------*
* END-OF-SELECTION
*---------------------------------------------------------------------*
END-OF-SELECTION.

*  报表程序终了处理
   PERFORM REPORT_EDIT_END.
*---------------------------------------------------------------------*
* TOP-OF-PAGE
*---------------------------------------------------------------------*
TOP-OF-PAGE.

*  报表页眉编辑
   PERFORM REPORT_HEADER_EDIT.
*---------------------------------------------------------------------*
* END-OF-PAGE
*---------------------------------------------------------------------*
   END-OF-PAGE.

*  报表页脚编辑
   PERFORM REPORT_FOOTER_EDIT.

*&---------------------------------------------------------------------*
*&          Form    CLASS_INPUT_CHECK
*&---------------------------------------------------------------------*
*           选择画面项目检查-班级
*---------------------------------------------------------------------*
FORM CLASSOO_INPUT_CHECK .
   TEST->ZZCLASS = P_CLASS.
   CALL METHOD TEST->GET_CLASS_DOMAIN.
   CALL METHOD TEST->GET_CLASS.

ENDFORM.                                  " CLASS_INPUT_CHECK
*&---------------------------------------------------------------------*
*&          Form    REPORT_DATA_INIT
```

```
*&---------------------------------------------------------------------*
*           报表数据初始化
*---------------------------------------------------------------------*
FORM REPORT_DATA_INIT.

* 内部表初始化
  CLEAR：      TEST->H_TEST.                            "成绩表头
  REFRESH：   TEST->TEST.                               "成绩表

ENDFORM.                              " REPORT_DATA_INIT
*&---------------------------------------------------------------------*
*&          Form    REPORT_DATA_SELECT
*&---------------------------------------------------------------------*
*           报表数据选择
*---------------------------------------------------------------------*
FORM REPORT_DATA_SELECT .

  TEST->ZZSTDNOF = S_STDNO-LOW.
  TEST->ZZSTDNOT = S_STDNO-HIGH.
  CALL METHOD TEST->GET_TEST
    EXPORTING
      CLASS = P_CLASS
      YEAR    = P_YEAR.

ENDFORM.                              " REPORT_DATA_SELECT
*&---------------------------------------------------------------------*
*&          Form    REPORT_DATA_EXE
*&---------------------------------------------------------------------*
*           报表数据打印
*---------------------------------------------------------------------*
FORM REPORT_DATA_EXE .

  CHECK TEST->TEST[] IS NOT INITIAL.

  LOOP AT TEST->TEST INTO TEST->H_TEST.  "成绩内部表
    CLEAR FLG-CLS_FRST.                     "班级表示清空
    AT NEW ZZCLASS.                         "新班级
      MOVE CNS-ON TO FLG-CLS_FRST.          "班级标示设 ON
    ENDAT .
    RESERVE 3    LINES.
*     报表明细编辑
    PERFORM EDIT_REPORT_DTL.

    AT END OF ZZCLASS .                     "班级最后
*      报表挂线
```

```
      PERFORM EDIT_REPORT_CLS.
    ENDAT.

    AT END OF ZZYEAR.
      NEW-PAGE.
    ENDAT.

  ENDLOOP.

ENDFORM.                               " REPORT_DATA_EXE
*&---------------------------------------------------------------------*
*&          Form    REPORT_EDIT_END
*&---------------------------------------------------------------------*
*           报表程序终了处理
*----------------------------------------------------------------------*
FORM REPORT_EDIT_END .

  CHECK TEST->TEST[] IS NOT INITIAL.
*  当前行号小于88 时
  IF SY-LINNO < LINE88 .
    SKIP TO LINE LINE88 .
    PERFORM REPORT_FOOTER_EDIT.
  ENDIF.

ENDFORM.                               " REPORT_EDIT_END
*&---------------------------------------------------------------------*
*&          Form    REPORT_HEADER_EDIT
*&---------------------------------------------------------------------*
*           报表页眉编辑
*----------------------------------------------------------------------*
FORM REPORT_HEADER_EDIT .
  SKIP 1 .
  WRITE：/030 TEXT-002.                         "成绩表
  WRITE：/016 TEXT-001 ,                        "学年
              020 CNS-COLON,
              022 TEST->H_TEST-ZZYEAR.
  WRITE：    030    TEXT-003,                   "班主任
             036 CNS-COLON,
             038 TEST->ZTNAMEM（6）,
             045 TEST->ZTNAMES（6）.
  WRITE：/015 SY-ULINE（052）  .
  WRITE：/015 SY-VLINE,
          017 TEXT-004,                         "班级
          021 SY-VLINE,
          025 TEXT-005,                         "学生
```

```
            038 SY-VLINE,
            040 TEXT-006,                    "语文
            045 SY-VLINE,
            047 TEXT-007,                    "数学
            052 SY-VLINE,
            054 TEXT-008,                    "自然
            059 SY-VLINE,
            061 TEXT-009,                    "思品
            066 SY-VLINE.
  WRITE：/015 SY-VLINE,
            021 SY-VLINE,
            038 SY-VLINE,
            045 SY-VLINE,
            052 SY-VLINE,
            059 SY-VLINE,
            066 SY-VLINE,
            015 SY-ULINE(052).

ENDFORM.                              " REPORT_HEADER_EDIT
*&---------------------------------------------------------------------*
*&        Form    EDIT_REPORT_DTL
*&---------------------------------------------------------------------*
*         报表明细编辑
*----------------------------------------------------------------------*
FORM EDIT_REPORT_DTL .

  WRITE：/015      SY-VLINE,
            021      SY-VLINE,
            023      TEST->H_TEST-ZZNAME(14),          "学生
            038      SY-VLINE,
            041（4）  TEST->H_TEST-ZZTENKOK,          "语文
            045      SY-VLINE,
            048（4）  TEST->H_TEST-ZZTENSAN,          "数学
            052      SY-VLINE,
            055（4）  TEST->H_TEST-ZZTENRIK,          "自然
            059      SY-VLINE,
            062（4）  TEST->H_TEST-ZZTENSHA,          "思品
            066      SY-VLINE.
  IF FLG-CLS_FRST    = CNS-ON.                        "班级开始
    WRITE：  017 TEST->H_TEST-ZZCLASS+0(1),
            018 CNS-PLUS ,
            019 TEST->H_TEST-ZZCLASS+1(1).
ENDFORM.                              " EDIT_REPORT_DTL
*&---------------------------------------------------------------------*
*&        Form    EDIT_REPORT_CLS
```

```
*&---------------------------------------------------------------------*
*           报表挂线
*---------------------------------------------------------------------*
FORM EDIT_REPORT_CLS .

*  成绩表挂线
   WRITE：/015 SY-VLINE,
            021 SY-VLINE,
            038 SY-VLINE,
            045 SY-VLINE,
            052 SY-VLINE,
            059 SY-VLINE,
            066 SY-VLINE,
            015 SY-ULINE(052).

ENDFORM.                          " EDIT_REPORT_CLS
*&---------------------------------------------------------------------*
*&          Form    REPORT_FOOTER_EDIT
*&---------------------------------------------------------------------*
*           报表页脚编辑
*---------------------------------------------------------------------*
FORM REPORT_FOOTER_EDIT .

   WRITE 030 SY-PAGNO.            "页数

ENDFORM.                          " REPORT_FOOTER_EDIT
```

文本元素编辑如图 1-1 所示。

Program ZYOO_TEST Active

Text Symbols | Selection Texts | List Headings

Sym	Text	dLen	mLen
001	学年	4	4
002	成 绩 表	10	10
003	班主任	6	6
004	班级	4	4
005	学 生	8	8
006	语文	4	4
007	数学	4	4
008	自然	4	4
009	思品	4	4
		0	0

图 1-1

选择文本元素编辑如图 1-2 所示。

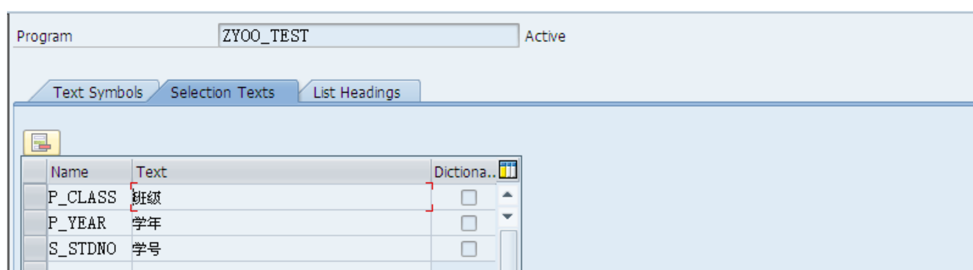


图 1-2

1.2.2 程序测试

运行程序结果如图 1-3 所示。

成绩表-面向对象

班级 1A

学年 2009

学号 00001 到 00009

图 1-3

填写相关参数，单击按钮，运行结果如图 1-4 所示。

成绩表-面向对象

成　绩　表

学年：2009　　班主任：刘备　　严长寿

班级	学　　生	语文	数学	自然	思品
1-A	郑喜定	60	68	70	81
	魏喜奎	62	65	74	83
	金喜善	62	65	71	89
	洪喜官	61	63	75	88
	朴喜珍	64	66	75	88
	王喜凤	64	66	75	88
	刘喜仁	64	66	76	88
	白喜医	62	66	72	88
	赵喜庆	62	63	72	82

图 1-4

1.3 全局类的实现

1.3.1 接口制作

（1）类制作器：第一画面

在“Objekttyp”文本框中输入接口的名称，单击 Anlegen 按钮，图 1-5 所示。

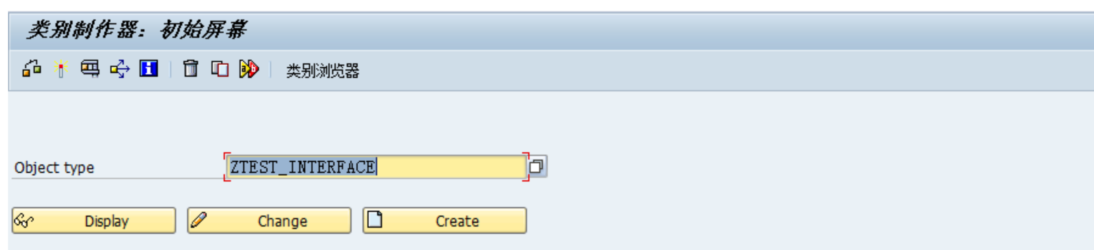


图 1-5

选中“界面”单选按钮，单击图 1-6 中的按钮。

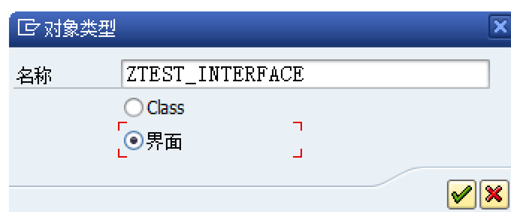


图 1-6

填写接口说明栏，单击图 1-7 中的 保存 按钮。

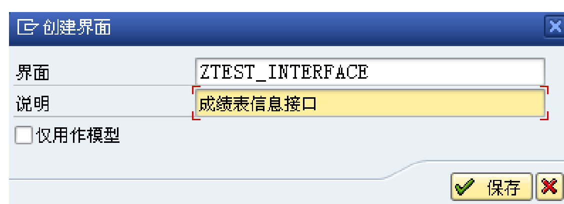


图 1-7

注：当**仅用作模型**复选框被勾选时，该接口不能被实例化，只能作为模型。

单击图 1-8 中的按钮。



图 1-8

单击图 1-9 中的✅按钮，由于接口初次创建，会弹出如图 1-10 所示信息。

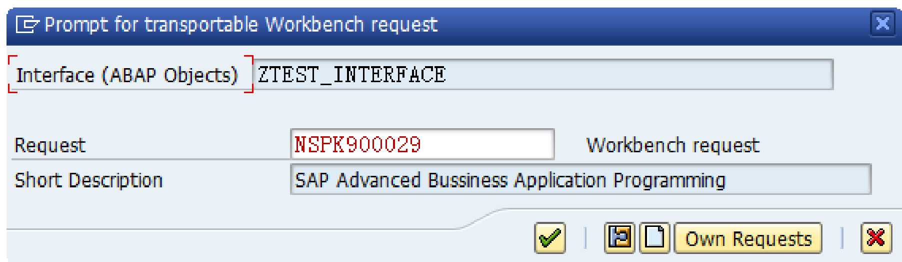


图 1-9



图 1-10

（2）类制作器：类编辑画面

单击编辑画面中各标签页，“属性”标签页编辑如图 1-11 所示。

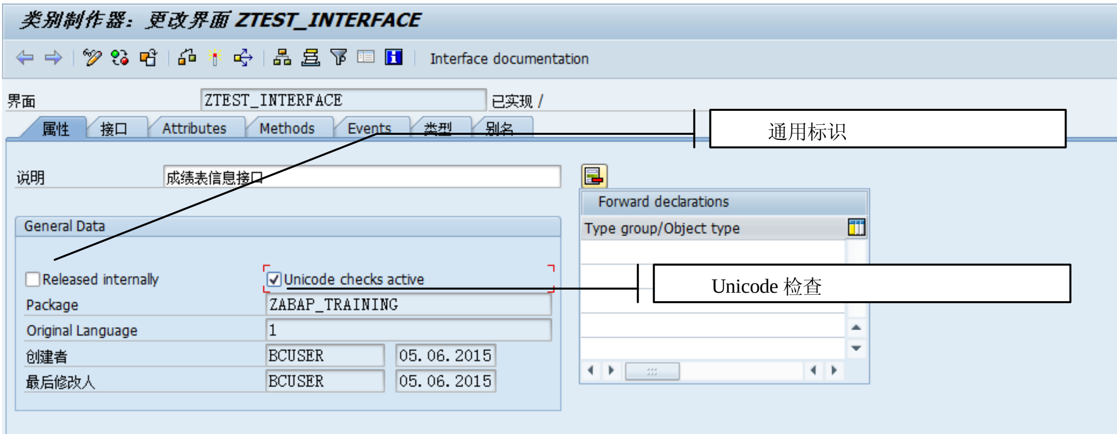


图 1-11

“接口”标签页编辑如图 1-12 所示。

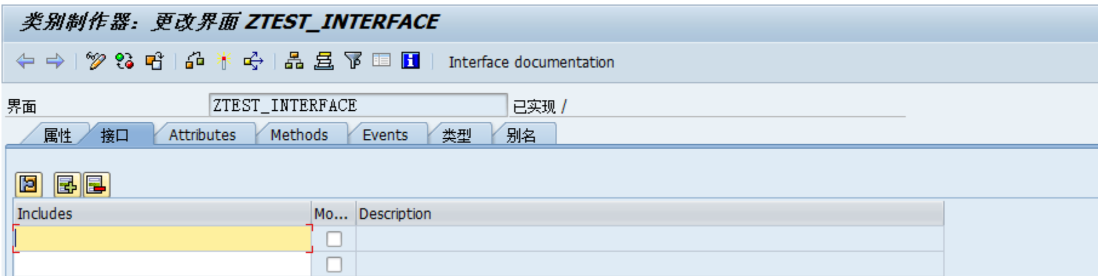


图 1-12

“Attribute”标签页编辑如图 1-13 所示。



图 1-13

“Methoden”标签页编辑如图 1-14 所示。

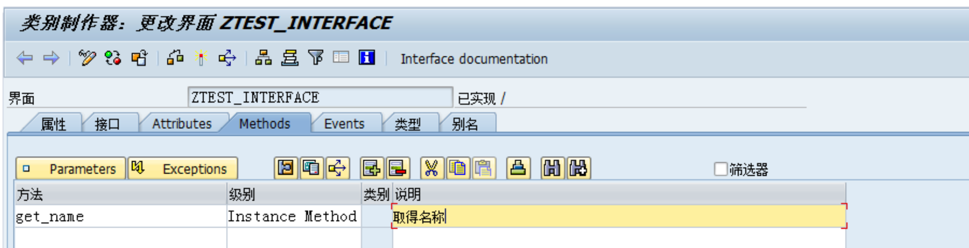


图 1-14

选中 get_name 方法，单击图 1-14 中的 Parameter 按钮，方法参数编辑如图 1-15 所示。

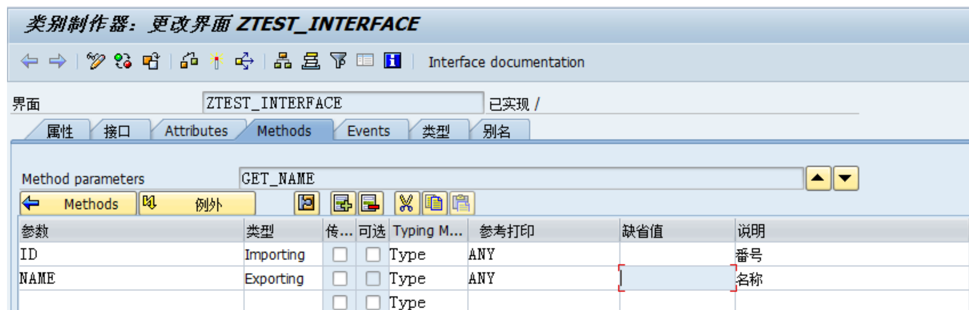


图 1-15

单击图 1-15 中的 例外 按钮，例外编辑如图 1-16 所示。

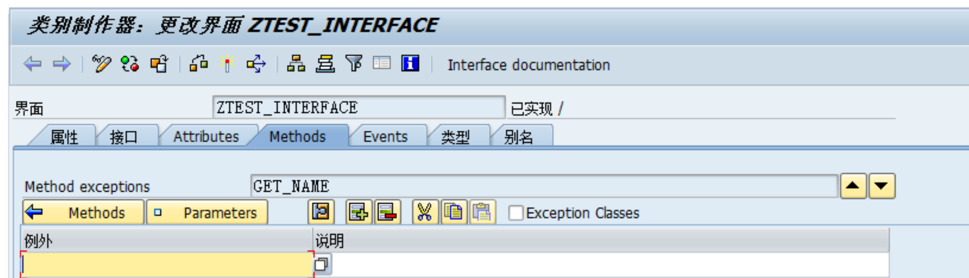


图 1-16

“Eregnisse”标签页编辑如图 1-17 所示。

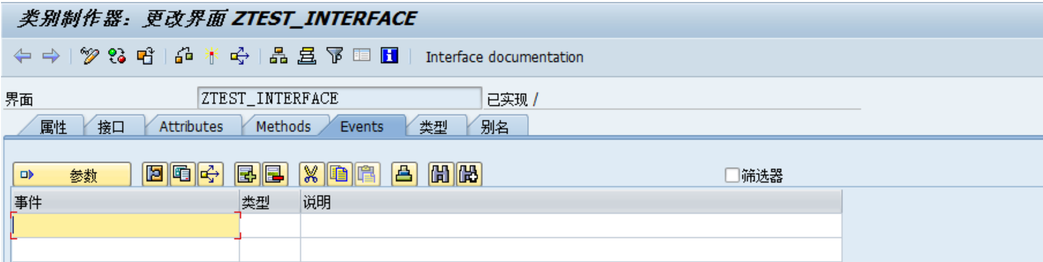


图 1-17

“类型”标签页编辑如图 1-18 所示。

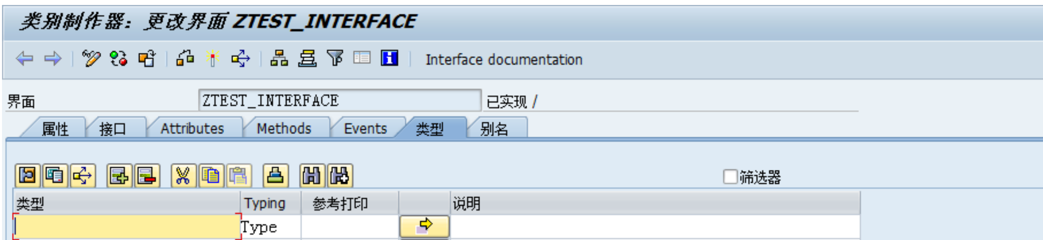


图 1-18

“别名”标签页编辑如图 1-19 所示。

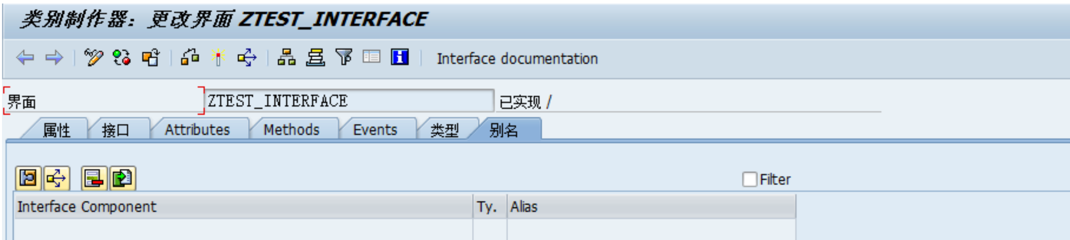


图 1-19

（3）类制作器：类编辑画面（接口的保存、检查及有效化）

单击标准工具栏中的按钮，效果如图 1-20 所示。



图 1-20

单击标准工具栏中的按钮，效果如图 1-21 所示。



图 1-21

单击标准工具栏中的按钮，效果如图 1-22 所示。



图 1-22

1.3.2 接口的使用（类的创建）

（1）类制作器：第一画面

在“Object type”文本框中输入类的名称，单击 Anlegen 按钮，如图 1-23 所示。

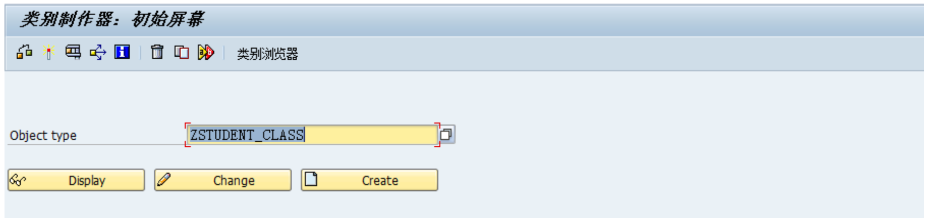


图 1-23

选中“Class”单选按钮，单击图 1-24 中的按钮。

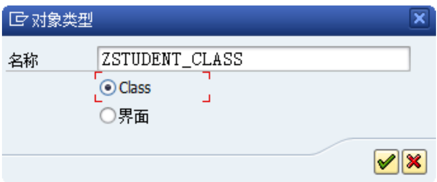


图 1-24

填写类说明栏，单击图 1-25 中的 保存 按钮。

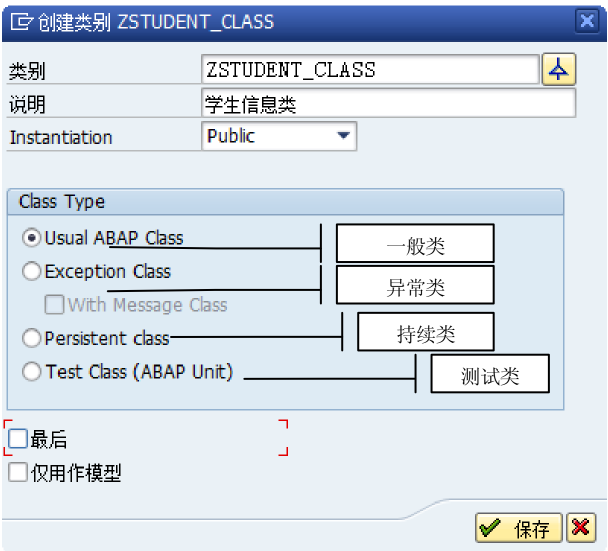


图 1-25

单击图 1-26 中的按钮。

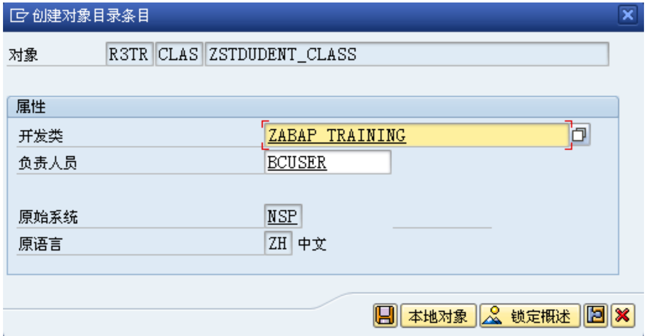


图 1-26

单击图 1-27 中的按钮。

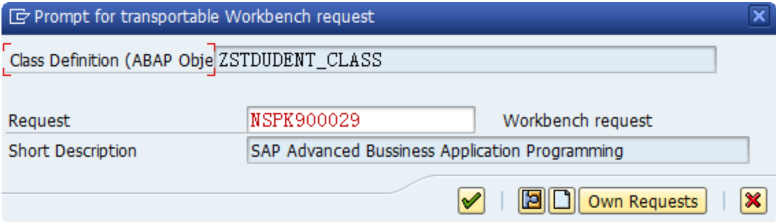


图 1-27

（2）类制作器：类编辑画面

单击编辑画面中各标签页，“Properties”标签页编辑如图 1-28 所示。

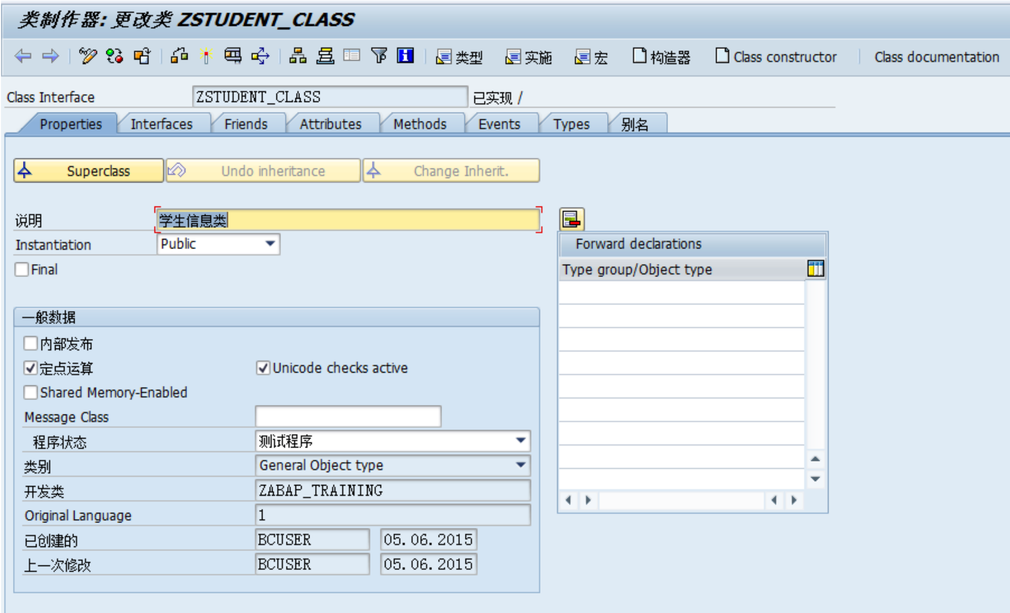


图 1-28

“Interfaces”标签页编辑如图 1-29 所示。

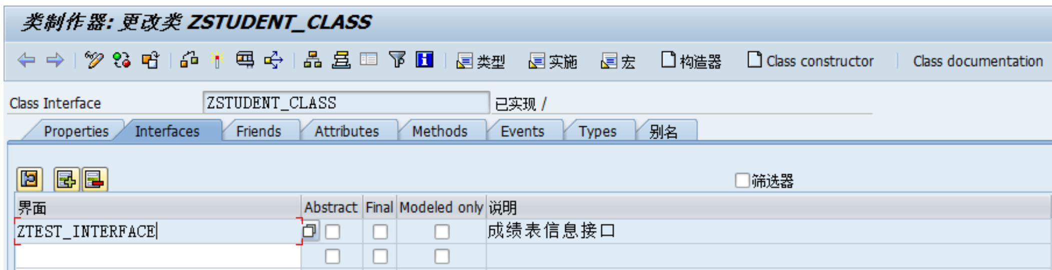


图 1-29

“Friends”标签页编辑如图 1-30 所示。

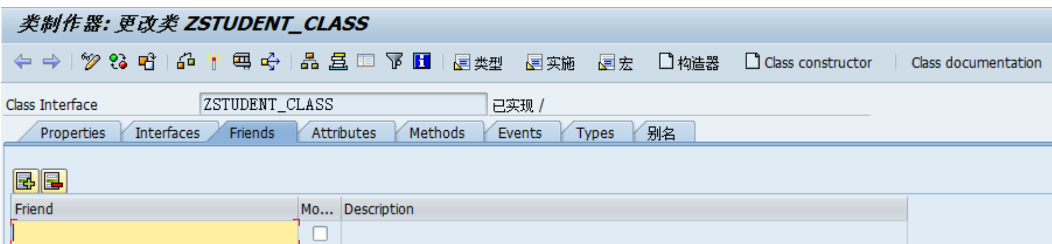


图 1-30

“Attributes”标签页编辑如图 1-31 所示。

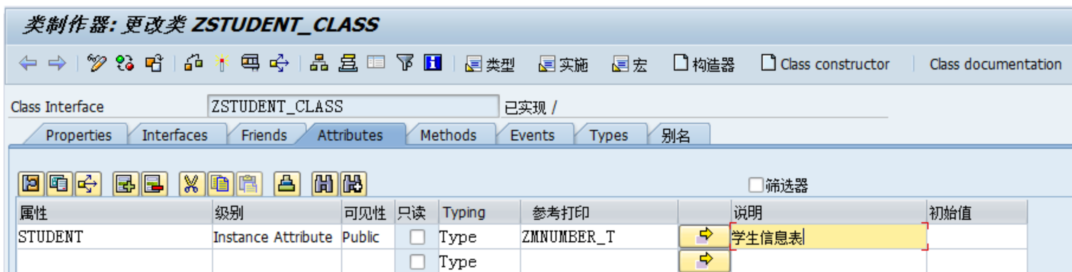


图 1-31

“Methods”标签页编辑如图 1-32 所示。

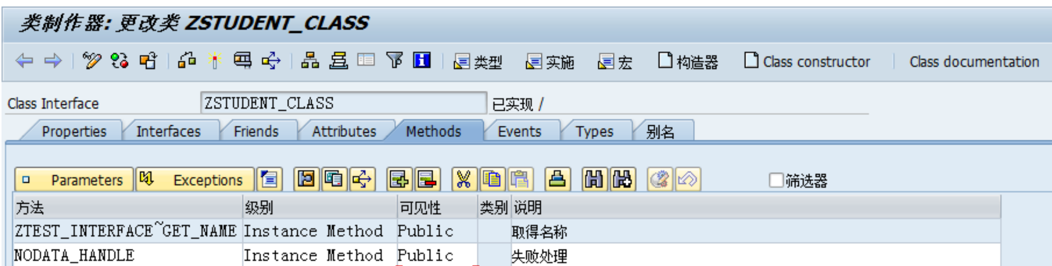


图 1-32

注意：由于上图中 NODATA_HANDLE 方法要用到 NODATA 事件，所以这里要先编辑事件标签页。

单击图 1-32 中的按钮，编辑代码如图 1-33 所示。

Class Builder: Class ZSTDUDENT_CLASS Change

Pattern　Pretty Printer　Signature　Public Section

Ty.	Parameter	Type spec.	Descri...
	VALUE(NAME)	TYPE ANY	名称
	VALUE(ID)	TYPE ANY	番号

Method　ZTEST_INTERFACE~GET_NAME　Activ

```
METHOD ZTEST_INTERFACE~GET_NAME.
  SELECT  SINGLE *
      FROM  ZMNUMBER_T
      INTO ME->STDUDENT
     WHERE  ZZNUMBER  = ID   .

  IF SY-SUBRC = 0.
    NAME = ME->STDUDENT-ZZNAME.
  ELSE.
    RAISE EVENT NODATA EXPORTING ID = ID.
  ENDIF.

ENDMETHOD.
```

图 1-33

单击标准工具栏中的按钮，效果如图 1-34 所示。

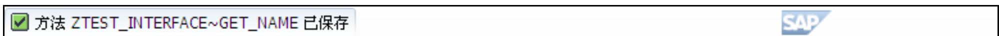


图 1-34

单击标准工具栏中的按钮，回到编辑画面，选中NODATA_HANDLE方法，单击按钮，打开图 1-35 中的窗口。

更改方法 NODATA_HANDLE

Object type	ZSTUDENT_CLASS
方法	NODATA_HANDLE
说明	失败处理

可见性：◉公共的　○被保护的　○私人的

方法：○Static　◉Instance

☐Abstract
☐Final
☑事件控制程序

类别/界面	ZSTUDENT_CLASS
事件	NODATA

☐模型化
☐编辑器锁定
☐活动的

Created
Last changed by (Defn)
Last changed by (Imp.)

更改

图 1-35

单击图 1-35 中的更改按钮，处理方法会绑定到事件中。“事件”列表框编辑如图 1-36 所示。

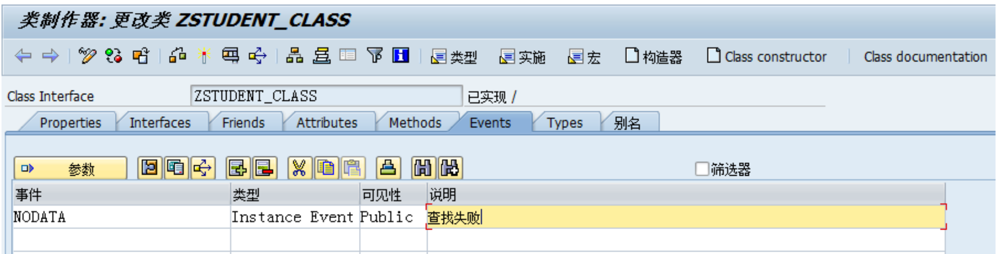


图 1-36

选中 NODATA 事件，单击图 1-36 中的参数按钮，编辑参数如图 1-37 所示。

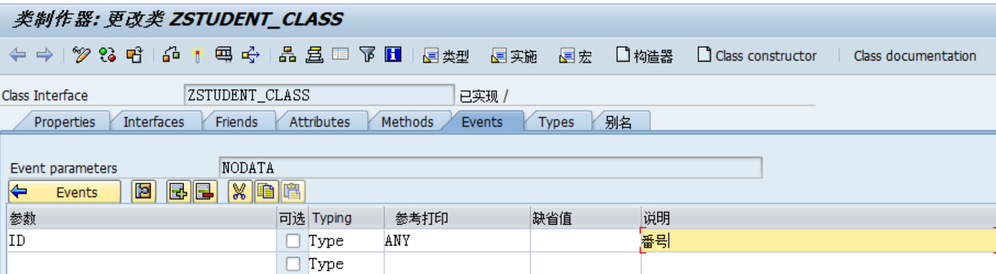


图 1-37

“Typen”标签页编辑如图 1-38 所示。

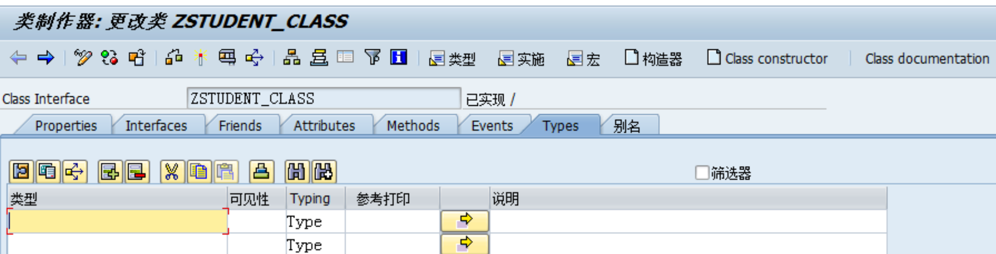


图 1-38

“别名”标签页编辑如图 1-39 所示。

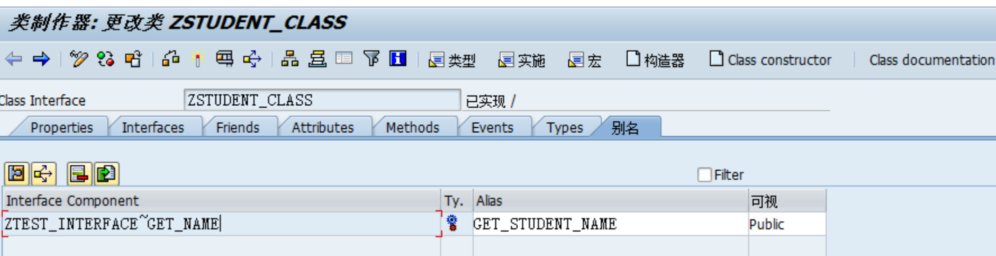


图 1-39

注：别名一般用来声明继承类中的方法名。

在“Methoden”标签页中，选中 NODATA_HANDLE 方法，单击图 1-39 中的 参数 按钮，编辑参数如图 1-40 所示。

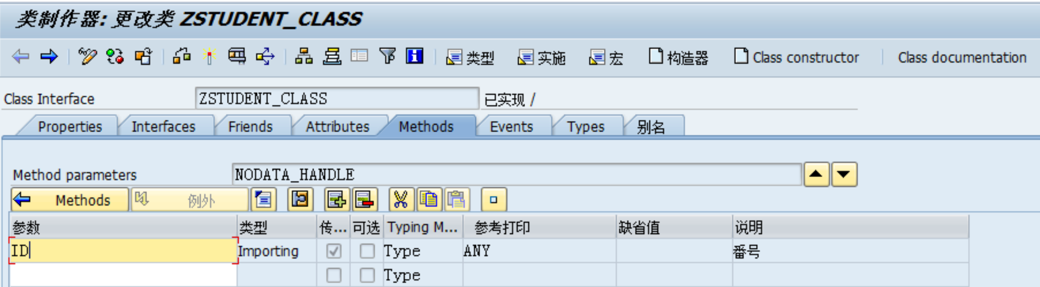


图 1-40

单击图 1-40 中的按钮，编辑代码如图 1-41 所示。

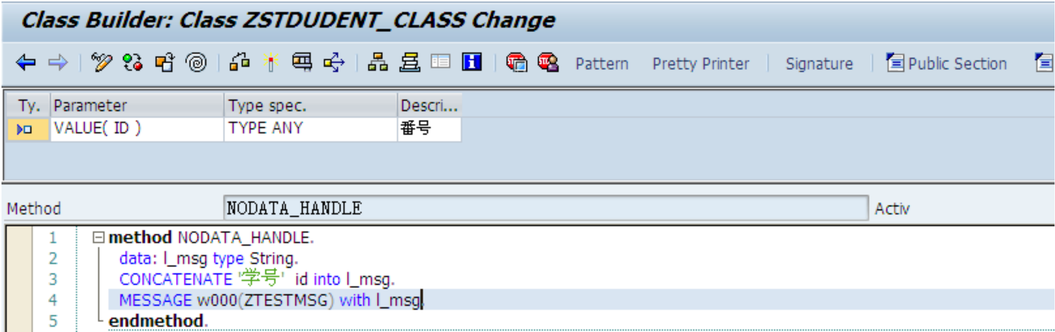


图 1-41

（3）类制作器：类编辑画面（类的保存、检查及有效化）

单击标准工具栏中的按钮，效果如图 1-42 所示。



图 1-42

单击标准工具栏中的按钮，效果如图 1-43 所示。



图 1-43

点击标准工具栏中的 按钮，效果如图 1-44 所示。



图 1-44

1.3.3 类的测试

类的测试语句如下：

```
DATA L_CLS_STUDENT TYPE REF TO ZSTDUDENT_CLASS.

CHECK TBL_TEST[] IS NOT INITIAL.
CREATE OBJECT L_CLS_STUDENT.
* 学生姓名取得
LOOP AT TBL_TEST INTO STR_TEST.
  L_CLS_STUDENT->ZTEST_INTERFACE～GET_NAME（
   EXPORTING
     ID        = STR_TEST-ZZSTDNO                  "学号
 IMPORTING
   NAME      = STR_TEST-ZZNAME                   "姓名
        ）.

  MODIFY TBL_TEST FROM STR_TEST.

ENDLOOP.
```

第 2 章　ALV应用

ALV 是 SAP List Viewer（也作 ABAP List Viewer），是 SAP 中常用的经典报表输出格式。常用的 ALV 种类有以下 6 个：

- ALV 表格（ALV Grid）。
- ALV 简单列表（ALV Simple List）。
- 多表格的 ALV 列表。
- 层级显示的 ALV 列表（Hierarchical-Sequential List）。
- 面向对象 ALV。
- ALV 树。

下面的内容将根据不同的 ALV 类型分开介绍。

1．ALV表格

ALV 表格是最常用的一种报表输出格式，如图 2−1 所示。

数据浏览器：表格 ZMNUMBER_T 选择条目　　12

检查表格...

MANDT	ZZNUMBER	ZZNAME	ZZSEX	ZZFNAME	ZZMNAME	ZZMOBILE	ZZADDR	ZZNOTE
800	00001	郑喜定	W	郑和和	王顺英英	12384765678	北京鲁迅路126号	
800	00002	魏喜奎	M	魏东亭	张娜英英	13987645679	北京鲁迅路129号	修改后
800	00003	金喜善	W	金魏亭	谢爽英英	13987645677	北京鲁迅路130号	修改后
800	00004	洪喜官	M	洪正哲	萧芳英英	13987645680	北京鲁迅路131号	修改后
800	00005	朴喜珍	M	朴仁勇	郝彤英英	13987645681	北京鲁迅路132号	修改后
800	00006	王喜凤	W	王烈	李志华	13987645682	北京鲁迅路133号	
800	00007	刘喜仁	M	刘秀	侯灿华	13987645683	北京鲁迅路134号	
800	00008	白喜医	M	白崇禧	王丹丹	13987645684	北京鲁迅路135号	
800	00009	赵喜庆	W	赵刚禧	马露莎	13987645685	北京鲁迅路136号	
800	00010	张喜早	M	张清禧	洪欣莎	13987645686	北京鲁迅路137号	
800	00011	刘喜位	M	刘流禧	王云莎	17269098876	大连市甘井子区百合山庄25号	
800	00012	郝喜联	M	郝文理	催花莎	13809076754	大连市甘井子区百合山庄25号	更新

图 2−1

（1）适用范围

- 标准功能强大，且外观上可直接用鼠标拖拽栏位的宽度和顺序，一般无特别要求的报表均采用此格式。
- 输出格式与 Excel 文档相似，当客户要求下载 Excel 文档时，多采用此输出格式。ALV 表格每个格只能显示不多于 128 个字符，不能多行显示 1 条数据。

（2）实现方式

- 构建字段目录（Field Catalog）表（输出表的表头栏位），一般采用函数‘REUSE_ALV_FIELDCATALOG_MERGE’取得。
- 设置字段目录表中各栏位的相关属性，一般需修改显示名称、显示长度、锁定、是否加总、是否显示等。
- 输出报表：Call Function　‘REUSE_ALV_GRID_DISPLAY’。

2．ALV简单列表（Simple List）

与 ALV 表格相比，除了报表自身在显示上的差别外，其他大致相同。ALV 列表更接近标准的一览输出，如图 2−2 所示。

数据浏览器：表格 ZMNUMBER_T 选择条目　　12

检查表格…

MANDT	ZZNUMBER	ZZNAME	ZZSEX	ZZFNAME	ZZMNAME	ZZMOBILE	ZZADDR	ZZNOTE
800	00001	郑喜定	W	郑和和	王顺英英	12384765678	北京鲁迅路126号	
800	00002	魏喜奎	M	魏东亭	张娜英英	13987645679	北京鲁迅路129号	修改后
800	00003	金喜善	W	金魏亭	谢爽英英	13987645677	北京鲁迅路130号	修改后
800	00004	洪喜官	M	洪正哲	萧芳英英	13987645680	北京鲁迅路131号	修改后
800	00005	朴喜珍	M	朴仁勇	郝彤英英	13987645681	北京鲁迅路132号	修改后
800	00006	王喜凤	W	王烈	李志华	13987645682	北京鲁迅路133号	
800	00007	刘喜仁	M	刘秀	侯灿华	13987645683	北京鲁迅路134号	
800	00008	白喜医	M	白崇禧	王丹丹	13987645684	北京鲁迅路135号	
800	00009	赵喜庆	W	赵刚禧	马露莎	13987645685	北京鲁迅路136号	
800	00010	张喜早	M	张清禧	洪欣莎	13987645686	北京鲁迅路137号	
800	00011	刘喜位	M	刘流禧	王云莎	17269098876	大连市甘井子区百合山庄25号	
800	00012	郝喜联	M	郝文理	催花莎	13809076754	大连市甘井子区百合山庄25号	更新

图 2−2

（1）适用范围

- 标准功能同 ALV 表格大致相同，栏位宽度已最优化显示，但不可直接用鼠标拖拽栏位的顺序，这不符合某些用户习惯，故使用范围不如 ALV 表格广泛，但当需要翻页时，只能采用 ALV 列表而不能用 ALV 表格。
- 输出格式一览类似，一般用于执行事务后返回消息的显示，可以多行显示 1 条数据，但只能显示不多于 90 个字段的表的数据，且排序的列相同的值不能被合并。

（2）实现方式

- 构建字段目录（Field Catalog）表，与 ALV 表格实现方法相同，一般采用函数‘REUSE_ALV_FIELDCATALOG_MERGE’得到。
- 设置字段目录表中各栏位的相关属性，一般需要修改显示名称、显示长度、锁定、是否加总、是否显示等。
- 输出报表：Call Function ‘REUSE_ALV_LIST_DISPLAY’。

3．多表格的ALV列表

区别于 ALV 简单列表只有一个表身，该种 ALV 可在一个画面上显示多个表格，如图 2−3 所示。

总账科目行项目显示

选择　争议事件

总账科目　70000　普通股
公司代码　1000

表头一

>	分配	凭证编号	部门	凭证类型	凭证日期	PK	本币金额	本币	税	清账凭证
✔	19950102	100007382	9900	SA	1995/01/02	50	12, 782, 297. 03-	EUR		
✔	19961201	100007559	9900	SA	1996/12/01	50	6, 051, 399. 35-	EUR		
✔	19970101	100007560	7000	SA	1997/01/01	50	95, 867. 23-	EUR		
✔	19970101	100007562	7000	AB	1997/01/01	40	95, 867. 23	EUR		
✔	19971212	100007869		SA	1997/12/12	50	979, 737. 88-	EUR		
✔	19971231	100007870		SA	1997/12/31	50	13, 702. 62-	EUR		
✔	19971231	100007871		AB	1997/12/31	40	13, 702. 62	EUR		
✔	19971231	100007872		SA	1997/12/31	40	13, 702. 62	EUR		
* ✔							19, 799, 731. 64-	EUR		
** 科目 70000							19, 799, 731. 64-	EUR		

总账科目　71000　资本储备
公司代码　1000

表头二

>	分配	凭证编号	部门	凭证类型	凭证日期	PK	本币金额	本币	税	清账凭证
✔	19951231	100007383	9900	SA	1995/12/31	50	7, 158, 086. 34-	EUR		
✔	19951231	100007555		SA	1995/12/31	50	1, 141, 754. 14-	EUR		

图 2−3

（1）适用范围

- 当一个报表包含多个表格时，需采用此种输出方式。
- ALV 简单列表需要注意的地方此种列表同样需要注意。

（2）实现方式

- 构建字段目录表，有几个不同的表要输出就构建几次。与 ALV 表格实现方法相同，一般采用函数‘REUSE_ALV_FIELDCATALOG_MERGE’，也可以手动构建。
- 设置字段目录表中各栏位的相关属性，一般需要修改显示名称、显示长度、锁定、是否加总、是否显示等；
- 初始化屏幕：调用函数‘REUSE_ALV_BLOCK_LIST_INIT’。添加数据：调用函数‘REUSE_ALV_BLOCK_LIST_APPEND’，有几个表就要调用几次。输出报表：调用函数'REUSE_ALV_BLOCK_LIST_DISPLAY’。

4．层级显示的ALV列表

层级显示的 ALV List 在 SAP 中称为层级显示列表，如图 2-4 所示。

```
物料的采购凭证
打印预览  采购订单历史  修改  交货计划  服务
                                                                表头
PO          类型  供应商      名称                         PGp 订单   日期
  项目   物料                 短文本                           物料组
  D I A  工厂  地点              订购数量    Un      净价      币种    per Un
4151508101 NB  T-K515A01  Sapsota Company Limited          001  2003/02/24
  00010                   Ordner A4                             006
      K 1000                       100   PC        2.00   EUR        1 PC
     仍要交货                       60   PC      120.00   EUR    60.00 %
     仍要开票       明细            100   PC      200.00   EUR   100.00 %
  00020                   Bleistifte                            006
      K 1000                       100   PC        0.50   EUR        1 PC
     仍要交货                         0   PC        0.00   EUR     0.00 %
     仍要开票                       100   PC       50.00   EUR   100.00 %
  00030                   Dokumentation                         006
      K 1000                         1   PC       40.00   EUR        1 PC
     仍要交货                         0   PC        0.00   EUR     0.00 %
     仍要开票                         1   PC       40.00   EUR   100.00 %
4151508102 NB  T-K515A02  Sapsota Company Limited          001  2003/02/24
  00010                   Ordner A4                             006
      K 1000                       100   PC        2.00   EUR        1 PC
     仍要交货                       60   PC      120.00   EUR    60.00 %
     仍要开票                       100   PC      200.00   EUR   100.00 %
  00020                   Bleistifte                            006
      K 1000                       100   PC        0.50   EUR        1 PC
     仍要交货                         0   PC        0.00   EUR     0.00 %
     仍要开票                       100   PC       50.00   EUR   100.00 %
```

图 2-4

（1）适用范围

这种形式的列表、表头和明细是由相关的字段联系起来的，表头和明细数据一对多。当然可以把表头和明细表分成两个列表在同一屏幕显示，但此种列表看起来更直观。所以当输出的报表数据有表头和明细的区分时，可以采用此种列表。

（2）实现方式一：Call Method（面向对象）

- 捆绑表头和明细表

SALV_T_HIERSEQ_BINDING（表头和明细相关的字段表类型）

SALV_S_HIERSEQ_BINDING（表头和明细相关的字段结构体）

● 得到输出表：调用方法 CL_SALV_HIERSEQ_TABLE=>FACTORY，取到的输出表对象（CL_SALV_HIERSEQ_TABLE）。

● 显示报表：调用方法 DISPLAY 输出表对象（CL_SALV_HIERSEQ_TABLE）->display（ ）。

（3）实现方式二：Call Function

● 构建表头和明细的栏位。

可以调用函数'REUSE_ALV_FIELDCATALOG_MERGE'，表头和明细的栏位分别构建后再追加到一个表里。也可以手动构建。

● 显示数据：调用函数'REUSE_ALV_HIERSEQ_LIST_DISPLAY'。

5．ALV树

ALV 树形结构可以显示多个层级，每个层级在相同的栏位名下显示；而之前的层级列表只能显示两个层级，而且两个层级之间要有主键去关联，数据显示在不同栏位名下，如图 2–5 所示。

SAP

成绩表

作成日 Today:2012/11/19 Time:14:44:49
选择基准 班级:1-A 学年:2009

作成者:YIDATEC

班级/学年/学生番号	语文...	数学...	自然...	思品...
1A	839	867	950	0
2009	839	867	950	0
00001	60	68	70	81
00002	62	65	74	83
00003	62	65	71	89
00004	61	63	75	88
00005	64	66	75	88
00006	64	66	75	88
00007	64	66	76	88
00008	62	66	72	88
00009	62	63	72	82

退出

图 2–5

（1）适用范围

● 一般在输出物料清单等层级结构比较明显的数据时使用，ALV 树每格最多支持 128 个字符。

● 不支持格合并，没有数据的过滤功能，系统标准没有下载功能，需要自己写代码。

● 只能显示 1000 个节点，只支持平面表，不支持多维表。

（2）实现方式（简单的 Tree 结构，在屏幕的 PBO 事件中处理）

● 创建并绘制屏幕和容器：容器可以不手动绘制，直接指定属性 CL_GUI_

CONTAINER=>SCREEN0，可全屏显示。

- 构建栏位名：CALL FUNCTION 'LVC_FIELDCATALOG_MERGE'。
- 构建排序表，对输出表进行层级设置。
- 实例化对象：CREATE OBJECT cl_gui_alv_tree_simple。
- 输出数据：CALL METHOD 实例化对象（cl_gui_alv_tree_simple）-> SET_TABLE_FOR_FIRST_DISPLAY。

6．面向对象ALV

面向对象 ALV 是 SAP 提供的 ALV 的对象模式（Object Model），即用面向对象技术对 ALV 进行封装，提供相应的工具和方式，如图 2-6 所示。。

成绩表

检查表格...

成绩表
作成日 Today:2012/11/19 Time:15:19:28
选择基准 班级:1-A 学年:2009
作成者: YIDATEC

班级	学年	学生...	学年	语文成	数学成	自然成	思品成
1A	2009	00001	郑喜定	60	68	70	81
		00002	魏喜峯	62	65	74	83
		00003	金喜善	62	65	71	89
		00004	洪喜官	61	63	75	88
		00005	朴喜珍	64	66	75	88
		00006	王喜凤	64	66	75	88
		00007	刘喜仁	64	66	76	88
		00008	白喜医	62	66	72	88
		00009	赵喜庆	62	63	72	82

图 2-6

（1）适用范围

面向对象的方式，功能更强大、更灵活。当用普通的 ALV 表格解决不了某些事件的问题时，大部分使用面向对象就可以解决。

（2）实现方式

1）框架（带控件）

- 在画面上定义一个定制控件并实例化对象 OBJ_WCL_CONTAINER。
- 生成一个 CL_GUI_CUSTOM_CONTAINER 类实例，在构造器参数中传入定制控件的名称。
- 构建栏位名：CALL FUNCTION 'LVC_FIELDCATALOG_MERGE'。
- 显示报表：CALL METHOD 类实例（CL_GUI_CUSTOM_CONTAINER）->set_table_for_first_display。

2）封装（不带控件）

a．首先要取得类 CL_SALV_FUNCTIONS 的实例，根据类 CL_SALV_TABLE 的方法 get_functions 可以取得功能实例。

```
lr_functions = gr_table->get_functions（ ）.
lr_functions->set_all( 'X' ).
```

b．根据类 CL_SALV_TABLE 的方法 set_screen_status 设置功能栏。

```
gr_table->set_screen_status（
    pfstatus        =   'SALV_STANDARD'
    report          =   sy-repid
    set_functions = gr_table->c_functions_all ）
```

c．自定义功能的实现

ALV 的自定义功能的实现动作要通过类的事件来处理。

● 首先定义事件的操作类。

```
CLASS lcl_handle_events DEFINITION.
  PUBLIC SECTION.
    CLASS-METHODS ： on_user_command FOR EVENT added_function OF cl_salv_events
IMPORTING e_salv_function.
ENDCLASS.                    "lcl_handle_events DEFINITION
```

● 实现事件的操作类。

```
CLASS lcl_handle_events IMPLEMENTATION.
  METHOD on_user_command.
    DATA l_string TYPE string.
    CONCATENATE 'Function is ' e_salv_function
            INTO l_string
            SEPARATED BY space.
    MESSAGE i000（0k） WITH l_string.
  ENDMETHOD.                    "on_user_command
ENDCLASS.                    "lcl_handle_events IMPLEMENTATION
```

● 注册事件。

```
SET HANDLER lcl_handle_events=>on_user_command FOR lr_events.
```

d．显示列表

```
gr_table->display（ ）
```

总结：以上是 ALV 类型的简单介绍，具体每种类型的详细用法，参照如下函数的详细说明。就 ALV 的形式和实现总结如下：

a．基于 ABAP 基本列表的 ALV。

● 简单表单实现：调用函数 REUSE_ALV_LIST_DISPLAY。
● 多表单实现：调用函数 REUSE_ALV_HIERSEQ_LIST_DISPLAY。
● 树形表单实现：调用函数 RS_TREE_LIST_DISPLAY。

b．基于 SAP 控制框架（Control Framework）。

● 函数调用实现（全画面模式）：调用函数 REUSE_ALV_GRID_DISPLAY 和

REUSE_ALV_GRID_DISPLAY_LVC。

- 面向对象实现： 使用类 CL_GUI_ALV_GRID 及 CL_GUI_ALV_TREE。

c．基于面向对象的封装。

- 通过封装调用以下类中的方法：CL_SALV_TABLE、 CL_SALV_HIERSEQ_TABLE 和 CL_SALV_TREE。

注：面向对象的方式创建 ALV 表格控件，是控制框架（Control Framework）中的一个组件，另外也可以采用 CL_SALV_TABLE、CL_SALV_HIERSEQ_TABLE、CL_SALV_TREE 等。后者创建可以是列表（List）形式，也可以是表格（Grid）形式，是目前 SAP 推荐的创建方式，是在 NetWeaver 2004 以后支持的版本。

全屏幕的表格控件有个对应的函数 REUSE_ALV_GRID_DISPLAY_LVC，支持面向对象的处理。面向对象的方式功能更强大一些，这种方式支持求最大值、最小值等标准功能。目前 SAP 中的组件已经以面向对象方式重写，采用面向对象方式来编写是以后的发展方向。

- 相关类:

CL_GUI_ALV_GRID：ALV 表格组件类，ALV List Viewer。

CL_GUI_CUSTOM_CONTAINER：画面中客户化控件的容器类。

- 相关类型:

LVC_T_FCAT：清单观察器控制的字段目录。

LVC_S_FCAT：ALV 控制的字段目录。

2.1 函数ALV和面向对象事件

2.1.1 程序设计

1．程序源码设计如下：

```
*---------------------------------------------------------------------*
*
* Program ID              ：ZREPO03_ALVOO_ADD
* Program Description     ：成绩查询ALV 显示
* Create    User          ：BCUSER
* Create    Date          ：2011/02/20
*
*---------------------------------------------------------------------*
* Function Overview
*  程序实现以班级，年度，学生为参照进行查询
*    技术点：FM-ALV 形式显示，视图应用，ALV 内部事件
*
*---------------------------------------------------------------------*
* Change List             ：
*      change NO.   Change Date    Change User    Change Detail
*      0000001      2011/02/20     BCUSER         程序创建
```

```
*------------------------------------------------------------------*

REPORT ZREPO03_ALVOO_ADD
   NO STANDARD PAGE HEADING
   LINE-SIZE 120
   LINE-COUNT 90（3）
   MESSAGE-ID ZTESTMSG.

*------------------------------------------------------------------*
* INCLUDE
*------------------------------------------------------------------*
INCLUDE：  ZTEST0N,     "常量定义
           ZTEST0F.     "Form 定义

*------------------------------------------------------------------*
* TYPE-POOLS
*------------------------------------------------------------------*
TYPE-POOLS SLIS.

*------------------------------------------------------------------*
* TYPES
*------------------------------------------------------------------*
*   成绩表
TYPES：  BEGIN OF TYP_TEST,
           SEL,                                          "选择
           ZZCLASS     LIKE ZTTEST_T-ZZCLASS,            "班级
           ZZYEAR      LIKE ZTTEST_T-ZZYEAR,             "学年
           ZZSTDNO     LIKE ZTTEST_T-ZZSTDNO,            "学号
           ZZNAME      LIKE ZMNUMBER_T-ZZNAME,           "姓名
           ZZTENKOK    LIKE ZTTEST_T-ZZTENKOK,           "语文
           ZZTENSAN    LIKE ZTTEST_T-ZZTENSAN,           "数学
           ZZTENRIK    LIKE ZTTEST_T-ZZTENRIK,           "自然
           ZZTENSHA    LIKE ZTTEST_T-ZZTENSHA,           "思想品德
         END     OF TYP_TEST.

TYPES   TYP_ITAB_TEST TYPE STANDARD TABLE OF TYP_TEST.

*------------------------------------------------------------------*
* TABLES：
*------------------------------------------------------------------*
TABLES：  ZTTEST_T,                           "考试成绩表
          ZMNUMBER_T.                         "学生信息表
* STATICS
*------------------------------------------------------------------*
*STATICS：  XX TYPE X VALUE XX.
```

```
*-------------------------------------------------------------------*
* CONSTANTS
*-------------------------------------------------------------------*
CONSTANTS ：
  BEGIN OF AC,
    PROGRAM_NAME     TYPE SY-REPID             VALUE 'ZREPO03',    "程序
*   控制
    STRUCTURE_NAME  TYPE DD02L-TABNAME     VALUE 'ZZTEST_S',       "结构体
    ZZCLASS              TYPE DD03L-FIELDNAME VALUE 'ZZCLASS',      "班级
    ZZYEAR                TYPE DD03L-FIELDNAME VALUE 'ZZYEAR',        "学年
    ZZSTDNO              TYPE DD03L-FIELDNAME VALUE 'ZZSTDNO',      "学号
    ZZTENKOK            TYPE DD03L-FIELDNAME VALUE 'ZZTENKOK',    "语文
    ZZTENSAN            TYPE DD03L-FIELDNAME VALUE 'ZZTENSAN',     "数学
    ZZTENRIK             TYPE DD03L-FIELDNAME VALUE 'ZZTENRIK',      "自然
    ZZTENSHA            TYPE DD03L-FIELDNAME VALUE 'ZZTENSHA',     "思品
    F2CODE（4）          TYPE C                        VALUE '&IC1',            "F2（双击）
    X（1）                  TYPE C                        VALUE 'X',
    H（1）                  TYPE C                        VALUE 'H',
    S（1）                  TYPE C                        VALUE 'S',
    A（1）                  TYPE C                        VALUE 'A',
  END OF AC.
*-------------------------------------------------------------------*
* DATA
*-------------------------------------------------------------------*
DATA：
  BEGIN OF TNAME,
    NAME1 LIKE ZMTNUMBER_T-ZTNAME,                                    "班主任
    NAME2 LIKE ZMTNUMBER_T-ZTNAME,                                    "副班主任
  END OF TNAME.

DATA：    TBL_TEST    TYPE TYP_ITAB_TEST,                              "成绩表
          STR_TEST    TYPE TYP_TEST.                                   "表头

DATA：    BEGIN OF AW,
    FIELDCAT          TYPE    LVC_T_FCAT,"SLIS_T_FIELDCAT_ALV ,        "字段属性
    FLDCAT_R          TYPE    SLIS_T_FIELDCAT_ALV ,                    "字段的内容
    FLDCAT_H          TYPE    SLIS_FIELDCAT_ALV,
    LAYOUT            TYPE    LVC_S_LAYO,"SLIS_LAYOUT_ALV,             "布局
    SORT_REC          TYPE    LVC_S_SORT,                              "排序记录
    SORTER            TYPE    LVC_T_SORT,                              "排序属性
    VARIANT           TYPE    DISVARIANT,                              "变式
    GRID_SET          TYPE    LVC_S_GLAY,                              "表格设定
    TITLE             TYPE    LVC_TITLE,                               "成绩表标题
    GRID_TITLE        TYPE    LVC_TITLE,                               "成绩表行标题
    EVENT             TYPE    SLIS_T_EVENT,                            "ALV 事件
    EXIT（1）          TYPE    C,
```

```
  END OF AW.
*  声明ALV Grid 类
DATA GR_GRID TYPE REF TO CL_GUI_ALV_GRID.
*---------------------------------------------------------------------*
*           CLASS CLS_EVENT_RECEIVER DEFINITION
*---------------------------------------------------------------------*
*           类CLS_EVENT_RECEIVER 定义
CLASS CLS_EVENT_RECEIVER DEFINITION.
  PUBLIC SECTION.
    METHODS：
      HANDLE_MODIFY FOR EVENT DATA_CHANGED_FINISHED OF CL_GUI_ALV_GRID
        IMPORTING E_MODIFIED ET_GOOD_CELLS.
ENDCLASS.                   "CLS_EVENT_RECEIVER DEFINITION

*---------------------------------------------------------------------*
*           CLASS CLS_EVENT_RECEIVER IMPLEMENTATION
*---------------------------------------------------------------------*
*           类CLS_EVENT_RECEIVER 实装
*---------------------------------------------------------------------*
CLASS CLS_EVENT_RECEIVER IMPLEMENTATION.
  METHOD   HANDLE_MODIFY.
    DATA STBL       TYPE LVC_S_STBL.       "刷新稳定性
    DATA LWA_CELL LIKE LINE OF ET_GOOD_CELLS.
*    设定初始值
    STBL-ROW = 'X'."
    STBL-COL = 'X'."
*    ALV 刷新
    CALL METHOD GR_GRID->REFRESH_TABLE_DISPLAY
      EXPORTING
        IS_STABLE = STBL.

  ENDMETHOD.                    "HANDLE_MODIFY
ENDCLASS.                   "LCL_EVENT_RECEIVER IMPLEMENTATION

*---------------------------------------------------------------------*
* PARAMETERS
*---------------------------------------------------------------------*
PARAMETERS：  P_VARI      TYPE DISVARIANT-VARIANT.       "表示布局（ALV 用）
PARAMETERS   P_CLASS     TYPE ZTTEST_T-ZZCLASS          "班级
                         OBLIGATORY .                   "必须

PARAMETERS P_YEAR TYPE ZTTEST_T-ZZYEAR OBLIGATORY.      "学年

*---------------------------------------------------------------------*
* SELECT-OPTIONS
*---------------------------------------------------------------------*
```

```
SELECT-OPTIONS     S_STDNO    FOR ZTTEST_T-ZZSTDNO
                             MATCHCODE OBJECT ZZMCSBID.

*----------------------------------------------------------------------*
* INITIALIZATION
*----------------------------------------------------------------------*
INITIALIZATION.

*  数据初始化
   PERFORM DATA_INIT
                 USING
                      P_CLASS            "班级
                      P_YEAR.            "学年
*----------------------------------------------------------------------*
* AT SELECTION SCREEN
*----------------------------------------------------------------------*
AT SELECTION-SCREEN.

*  选择画面项目检查-班级
   PERFORM CLASS_INPUT_CHECK USING P_CLASS TEXT-003 CHANGING TNAME.

*  选择画面项目检查-学年
   PERFORM YEAR_INPUT_CHECK USING P_YEAR TEXT-004.

*----------------------------------------------------------------------*
* AT SELECTION SCREEN ON VALUE-REQUEST
*----------------------------------------------------------------------*
AT SELECTION-SCREEN ON VALUE-REQUEST FOR    P_VARI .

*  选择画面项目选择-布局
   PERFORM VARIANT_INPUT_SELECT.

*----------------------------------------------------------------------*
*START-OF-SELECTION
*----------------------------------------------------------------------*
START-OF-SELECTION.

*  报表数据初始化
   PERFORM REPORT_DATA_INIT.
*  报表数据选择
   PERFORM REPORT_DATA_SELECT.
*  报表数据显示
   PERFORM REPORT_DATA_LIST.

*----------------------------------------------------------------------*
* END-OF-SELECTION
```

```
*----------------------------------------------------------------------*
*END-OF-SELECTION.

*  报表程序终了处理
*    PERFORM REPORT_EDIT_END.
*----------------------------------------------------------------------*
* TOP-OF-PAGE
*----------------------------------------------------------------------*
*TOP-OF-PAGE.

*  报表页眉编辑
*    PERFORM REPORT_HEADER_EDIT.
*----------------------------------------------------------------------*
* END-OF-PAGE
*----------------------------------------------------------------------*
*END-OF-PAGE.

*  报表页脚编辑
*    PERFORM REPORT_FOOTER_EDIT.
*----------------------------------------------------------------------*
* AT LINE-SELECTION
*----------------------------------------------------------------------*
*AT LINE-SELECTION.

*  报表选择处理
*    PERFORM REPORT_LIST_SELECT.

*----------------------------------------------------------------------*
* AT USER-COMMAND
*----------------------------------------------------------------------*
*AT USER-COMMAND.

*  用户交互处理
*    PERFORM REPORT_USER_COMMAND.

*----------------------------------------------------------------------*
* AT PFNN
*----------------------------------------------------------------------*
* AT PFNN.

*  功能键响应处理
*       PERFORM REPORT_PFNN_COMMAND.
*&---------------------------------------------------------------------*
*&          Form    VARIANT_INPUT_SELECT
*&---------------------------------------------------------------------*
*           选择画面项目选择
*----------------------------------------------------------------------*
```

```
FORM VARIANT_INPUT_SELECT .
  AW-VARIANT-REPORT = SY-REPID.                                    "程序名
  CALL FUNCTION 'REUSE_ALV_VARIANT_F4'
    EXPORTING
      IS_VARIANT              = AW-VARIANT
      I_SAVE                  = AC-A
*     it_default_fieldcat =
    IMPORTING
      E_EXIT                  = AW-EXIT
      ES_VARIANT              = AW-VARIANT
    EXCEPTIONS
      NOT_FOUND = 2.
  IF  （ SY-SUBRC = 2 ）.
    MESSAGE ID SY-MSGID TYPE CNS-MTY_S    NUMBER SY-MSGNO
            WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ELSE.
    IF ( AW-EXIT = SPACE ).
      P_VARI = AW-VARIANT-VARIANT.
    ENDIF.
  ENDIF.

ENDFORM.                          " VARIANT_INPUT_SELECT
*&---------------------------------------------------------------------*
*&        Form    REPORT_DATA_INIT
*&---------------------------------------------------------------------*
*           报表数据初始化
*---------------------------------------------------------------------*
FORM REPORT_DATA_INIT .

* 内部表初始化
  CLEAR：      STR_TEST.                              "成绩表头
  REFRESH：    TBL_TEST.                              "成绩表
  MOVE：        TEXT-001 TO AW-TITLE ,                "标题
              TEXT-002 TO AW-GRID_TITLE .             "行标题

ENDFORM.                          " REPORT_DATA_INIT
*&---------------------------------------------------------------------*
*&        Form    REPORT_DATA_SELECT
*&---------------------------------------------------------------------*
*           报表数据选择
*---------------------------------------------------------------------*
FORM REPORT_DATA_SELECT .

* 成绩表数据抽取
  SELECT        ZZCLASS                     "班级
                ZZYEAR                      "学年
```

```
                ZZSTDNO                 "学号
                ZZNAME                  "姓名
                ZZTENKOK                "语文
                ZZTENSAN                "数学
                ZZTENRIK                "自然
                ZZTENSHA                "思品
      FROM ZZTEST_V                     "成绩表
      INTO CORRESPONDING FIELDS OF TABLE TBL_TEST        "成绩内部表
      WHERE ZZCLASS    =    P_CLASS     "班级
      AND      ZZYEAR = P_YEAR          "学年
      AND       ZZSTDNO    IN S_STDNO.  "学号
*  无数据情况
   IF SY-SUBRC <> 0.
      MESSAGE S002.
      STOP.
   ENDIF.

*  排序
   SORT TBL_TEST BY ZZCLASS ASCENDING          "班级
                    ZZYEAR ASCENDING           "学年
                    ZZSTDNO ASCENDING.         "学号

ENDFORM.                          " REPORT_DATA_SELECT
*&---------------------------------------------------------------------*
*&           Form    REPORT_DATA_LIST
*&---------------------------------------------------------------------*
*            报表数据显示
*----------------------------------------------------------------------*
FORM REPORT_DATA_LIST .

*  声明 Field Catalog 结构体
   DATA LS_FLDCAT_R LIKE LINE OF AW-FLDCAT_R.
   DATA LS_FIELDCAT LIKE LINE OF AW-FIELDCAT.
*  声明回调事件结构体
   DATA LS_EVENT LIKE LINE OF AW-EVENT.
*  生成 Field Catalog，
*  功能
*  用来获得程序中的内表  或者  DDIC 表的 Field Catalog
   CALL FUNCTION 'REUSE_ALV_FIELDCATALOG_MERGE'
      EXPORTING
         I_PROGRAM_NAME            = SY-REPID"PGMID「ZREPO03」
         I_INTERNAL_TABNAME        = 'TBL_TEST'         "设定内部表名
         I_STRUCTURE_NAME          = AC-STRUCTURE_NAME "参照结构体「ZZTEST_S」
*        I_CLIENT_NEVER_DISPLAY     = AC-X               "Client 隐藏
*        I_INCLNAME                 =                    "内部表中定义的 Include 名
         I_BYPASSING_BUFFER         = AC-X               "既存的缓存值无效
```

```
*       I_BUFFER_ACTIVE              =                         "激活缓存
      CHANGING
        CT_FIELDCAT                  = AW-FLDCAT_R       "有文本字段的目录
      EXCEPTIONS
        INCONSISTENT_INTERFACE  = 1
        PROGRAM_ERROR                = 2
        OTHERS                   = 3
              .
    IF ( SY-SUBRC <> 0 ).
      MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
                    WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
      EXIT.
    ELSE.
*     编辑 Field Catalog
      LOOP AT AW-FLDCAT_R INTO LS_FLDCAT_R.
        MOVE-CORRESPONDING    LS_FLDCAT_R TO LS_FIELDCAT.
        IF    LS_FIELDCAT-FIELDNAME = AC-ZZTENKOK       OR        "语文
              LS_FIELDCAT-FIELDNAME = AC-ZZTENSAN       OR        "数学
              LS_FIELDCAT-FIELDNAME = AC-ZZTENRIK       OR        "自然
              LS_FIELDCAT-FIELDNAME = AC-ZZTENSHA      .          "思品.
*           设定编辑和合计属性
          LS_FIELDCAT-EDIT      = AC-X.
          LS_FIELDCAT-DO_SUM = AC-X.

        ENDIF.
        APPEND LS_FIELDCAT TO AW-FIELDCAT .
      ENDLOOP.
    ENDIF.
*
* 编辑回调事件内部表
    LS_EVENT-NAME = 'CALLER_EXIT'.
    LS_EVENT-FORM = 'FRM_CALLER_EXIT_AT_START'.
    APPEND LS_EVENT TO AW-EVENT.

*-- 排序字段目录编辑集
*   CLEAR FLDCAT_R.    FLDCAT_R-KEY = ''.
*   MODIFY FIELDCAT FROM    FLDCAT_R TRANSPORTING KEY WHERE KEY = 'X'.
```

```
**--默认排序字段设定=> SORTER
*
  CLEAR： AW-SORT_REC, AW-SORTER.
  AW-SORT_REC-SPOS        = '1'.                "排序基准「第一」
  AW-SORT_REC-FIELDNAME = AC-ZZCLASS.          "参照字段名「ZZCLASS」
*   AW-SORT_REC-TABNAME     = AC-STRUCTURE_NAME."参照结构体「ZZTEST_S」
  AW-SORT_REC-UP          = AC-X .              "升序
* AW-SORT_REC-DOWN        = AC-X .              "降序
  APPEND AW-SORT_REC TO AW-SORTER.
*
  AW-SORT_REC-SPOS        = '2'.                "排序基准「第二」
  AW-SORT_REC-FIELDNAME = AC-ZZYEAR.           "参照字段名「ZZYEAR」
*   AW-SORT_REC-TABNAME     = AC-STRUCTURE_NAME. "参照结构体「ZZTEST_S」
  AW-SORT_REC-UP          = AC-X .              "升序
* AW-SORT_REC-DOWN        = AC-X .              "降序
  APPEND AW-SORT_REC TO AW-SORTER.
*
  AW-SORT_REC-SPOS        = '3'.                "排序基准「第三」
  AW-SORT_REC-FIELDNAME = AC-ZZSTDNO.          "参照字段名「ZZSTDNO」
*   AW-SORT_REC-TABNAME     = AC-STRUCTURE_NAME."参照结构体「ZZTEST_S」
  AW-SORT_REC-UP          = AC-X .              "升序
* AW-SORT_REC-DOWN        = AC-X .              "降序
  APPEND AW-SORT_REC TO AW-SORTER.
*
*--布局设定=> LAYOUT
  CLEAR AW-LAYOUT.
*   AW-LAYOUT-F2CODE                 = AC-F2CODE.      "F2 机能（双击）「POP」
*   AW-LAYOUT-COLWIDTH_OPTIMIZE = AC-X .          "列宽属性设定
  AW-LAYOUT-STYLEFNAME = 'STYLE'.
  AW-LAYOUT-SEL_MODE = 'D'.
  AW-LAYOUT-BOX_FNAME = 'SEL'.
  AW-LAYOUT-EXCP_LED = 'X'.    "ALV 控制： 例外作为 LED

*   AW-LAYOUT-EDIT                  = AC-X .              "编集状态设定
*
**--表格设定（TOP-PAGE 等输出设定）
* AW-GRID_SET-COLL_END_L         = AC-X .              "页脚初期输出宽度 0 %
```

```
* AW-GRID_SET-COLL_TOP_P           = AC-X .                   "页眉初期输出宽度 0 %
*
**--变式设定
  AW-VARIANT-REPORT                   = SY-REPID.           "程序 ID.
*
*  功能
*此功能模块把任何结构的内部表 ，
*输出在格式化的单一或多行申请表格。
*
*过程:
* #传递内部表，包含输出信息。
* #传递一个结构，它包含该列表的布局设置的总体布局。
* #内部表中的一个目录项的形式获得通过。
* #项目目录，写一个输出项目清单
*
  CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY_LVC'
   EXPORTING
*        I_INTERFACE_CHECK                 = ' '"  接口的一致性检查日志输出
*        I_BYPASSING_BUFFER                =   "  既存的缓存值无效
*        I_BUFFER_ACTIVE                   = AC-X                    "  激活缓存
       I_CALLBACK_PROGRAM                = SY-REPID                "  本程序的程序名
       I_CALLBACK_PF_STATUS_SET          = 'STATUS_SET'            " GUI Status 设定 Form
       I_CALLBACK_USER_COMMAND           = 'USER_COMMAND'          "  用户交互 Form
       I_CALLBACK_TOP_OF_PAGE            = 'TOP_OF_PAGE'           "  页眉编辑 Form
*        I_CALLBACK_HTML_TOP_OF_PAGE = ' '    " HTML TOP-OF-PAGE  用  EXIT Form
*        I_CALLBACK_HTML_END_OF_LIST = ' '    " HTML END-OF-LIST  用  EXIT Form
       I_STRUCTURE_NAME                        = AC-STRUCTURE_NAME     " DDIC 结构体
*        I_BACKGROUND_ID                         = ' ' "  图片对象 ID
       I_GRID_TITLE                            = AW-GRID_TITLE          "  标题
       I_GRID_SETTINGS                         = AW-GRID_SET            "  表格设定
       IS_LAYOUT_LVC                           = AW-LAYOUT              "  布局样式
       IT_FIELDCAT_LVC                         = AW-FIELDCAT            "  字段目录
*        IT_EXCLUDING                            =                        "  无效的机能代码
*        IT_SPECIAL_GROUPS_LVC                   =                        "列选择的字段组
       IT_SORT_LVC                             =    AW-SORTER           "  初期排序设定
*        IT_FILTER_LVC                           =   "  过滤器设定
*        IT_HYPERLINK                            = " ALV  超级链接
```

```
*       IS_SEL_HIDE                         =    " 隐藏选择按钮
        I_DEFAULT                           = AC-X             " 变式初期设定
        I_SAVE                              = AC-A             " 变式保存制限
        IS_VARIANT                          = AW-VARIANT       " 变式
        IT_EVENTS                           =    AW-EVENT      " 实行事件内部表
*       IT_EVENT_EXIT                       =    " 标准机能代码退出依赖表
*       IS_PRINT_LVC                        =    " 印刷
*       IS_REPREP_ID_LVC                    =       "Re/Re 接口初期化键值
*       I_SCREEN_START_COLUMN               = 1                " 弹出式报表坐标
*       I_SCREEN_START_LINE                 = 1                " 弹出式报表坐标
*       I_SCREEN_END_COLUMN                 = 90               " 弹出式报表坐标
*       I_SCREEN_END_LINE                   = 70               " 弹出式报表坐标
*       I_HTML_HEIGHT_TOP                   =
*       I_HTML_HEIGHT_END                   =
*       IT_EXCEPT_QINFO_LVC                 =
*       IR_SALV_FULLSCREEN_ADAPTER          =
*     IMPORTING
*       E_EXIT_CAUSED_BY_CALLER             =
*       ES_EXIT_CAUSED_BY_USER              =
      TABLES
        T_OUTTAB                            = TBL_TEST         "成绩内容
     EXCEPTIONS
        PROGRAM_ERROR                       = 1
        OTHERS                              = 2
                .
  IF SY-SUBRC <> 0.
    MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
                WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ENDIF.


ENDFORM.                    " REPORT_DATA_LIST
*&---------------------------------------------------------------------*
*&      Form   STATUS_SET
*&---------------------------------------------------------------------*
*         Gui Status 设定
*----------------------------------------------------------------------*
```

```
FORM STATUS_SET USING EXTAB TYPE SLIS_T_EXTAB.
  SET PF-STATUS 'ZTEST_ALV'.
ENDFORM.                                        "STATUS_SET
*&---------------------------------------------------------------------*
*&          Form    USER_COMMAND
*&---------------------------------------------------------------------*
*           用户处理
*----------------------------------------------------------------------*
FORM USER_COMMAND USING RF_UCOMM TYPE SY-UCOMM
                                                SELFIELD TYPE SLIS_SELFIELD.
  DATA：  MSGTXT（90）  TYPE C,
          MSGIDX（03）  TYPE N.
*
*  从 ALV 实例出来的数据更新 ALV 对应的内表
  CALL METHOD GR_GRID->CHECK_CHANGED_DATA.

  CASE RF_UCOMM.
*     双击事件
    WHEN AC-F2CODE.
      READ TABLE TBL_TEST INTO STR_TEST INDEX SELFIELD-TABINDEX.
      MSGIDX = SELFIELD-TABINDEX.
      CONCATENATE '行='              MSGIDX
                        '、项目名='      SELFIELD-FIELDNAME
                        '、项目内容=' SELFIELD-VALUE
              INTO    MSGTXT.
      MESSAGE S100（ZY）  WITH MSGTXT.
      SELFIELD-REFRESH       = AC-X .          "报表刷新
      SELFIELD-COL_STABLE = AC-X .          "光标位置
      SELFIELD-ROW_STABLE = AC-X .          "同上
  ENDCASE.
ENDFORM.                                          "USER_COMMAND
*&---------------------------------------------------------------------*
*&          Form    TOP_OF_PAGE
*&---------------------------------------------------------------------*
*           页眉编辑
*----------------------------------------------------------------------*
FORM TOP_OF_PAGE.
```

```
DATA：
  LISTHEADER       TYPE SLIS_T_LISTHEADER,
  WA_LISTHEADER TYPE SLIS_LISTHEADER,
  H_DATUM（10）,
  H_UZEIT（10）,
  EMPTY（3）          TYPE C VALUE '    '.
CLEAR WA_LISTHEADER.
* 标题
WA_LISTHEADER-TYP       = AC-H."'H'.
WA_LISTHEADER-INFO      = AW-TITLE.
IF ( NOT AW-VARIANT-TEXT IS INITIAL ).
  CONCATENATE AW-TITLE '（' AW-VARIANT-TEXT '）'
            INTO WA_LISTHEADER-INFO.
ENDIF.
APPEND WA_LISTHEADER TO LISTHEADER.
*
CLEAR WA_LISTHEADER.
APPEND WA_LISTHEADER TO LISTHEADER.
* 中标题
WRITE SY-DATUM TO H_DATUM DD/MM/YYYY.
WRITE SY-UZEIT TO H_UZEIT USING EDIT MASK '__：__：__'.

CLEAR WA_LISTHEADER.
WA_LISTHEADER-TYP       = AC-S."'S'.
WA_LISTHEADER-KEY       =   '作成日'.
CONCATENATE 'Today：' H_DATUM    EMPTY
                       'Time：'    H_UZEIT
            INTO WA_LISTHEADER-INFO .
APPEND WA_LISTHEADER TO LISTHEADER.

CLEAR WA_LISTHEADER.
WA_LISTHEADER-TYP       = AC-S."'S'.
WA_LISTHEADER-KEY       =   '选择基准'.
CONCATENATE '班级：' P_CLASS+0（1） CNS-PLUS     P_CLASS+1（1）    EMPTY
                       '学年：' P_YEAR
            INTO WA_LISTHEADER-INFO .
APPEND WA_LISTHEADER TO LISTHEADER.
```

```
* 小标题
  CLEAR WA_LISTHEADER.
  WA_LISTHEADER-TYP       = AC-A.
  CONCATENATE '作成者：' SY-UNAME    INTO WA_LISTHEADER-INFO .
  APPEND WA_LISTHEADER TO LISTHEADER.
* 页眉显示
  CALL FUNCTION 'REUSE_ALV_COMMENTARY_WRITE'
    EXPORTING
      IT_LIST_COMMENTARY = LISTHEADER.
ENDFORM.                              "TOP_OF_PAGE
*&---------------------------------------------------------------------*
*&          Form    F_CALLER_EXIT_AT_START
*&---------------------------------------------------------------------*
*           text
*----------------------------------------------------------------------*
*          -->RS_DATA       text
*----------------------------------------------------------------------*
FORM FRM_CALLER_EXIT_AT_START USING RS_DATA TYPE SLIS_DATA_CALLER_EXIT.
* 声明事件类
  DATA LR_EVENT TYPE REF TO CLS_EVENT_RECEIVER.
* 取得 ALV 实体
  CALL FUNCTION 'GET_GLOBALS_FROM_SLVC_FULLSCR'
    IMPORTING
      E_GRID = GR_GRID.
* 注册 ALV 事件
  CALL METHOD GR_GRID->REGISTER_EDIT_EVENT
    EXPORTING
      I_EVENT_ID = CL_GUI_ALV_GRID=>MC_EVT_ENTER
      ERROR           = 1
      OTHERS          = 2.
* 创建事件实体
  CREATE OBJECT LR_EVENT.
* 注册 ALV 事件处理程序
  SET HANDLER LR_EVENT->HANDLE_MODIFY FOR GR_GRID.
ENDFORM.                              "F_CALLER_EXIT_AT_START
```

文本元素编辑如图 2−7 所示。

Program ZREP003_ALV00_ADD Active

Text symbols | Selection texts | List Headings

Sym	Text	dLen	mLen
001	成绩表	6	6
002	成绩表行标题	12	12
003	班级	4	4
004	学年	4	4
		0	
		0	

图 2-7

选择文本元素编辑如图 2-8 所示。

Program ZREP003_ALV00_ADD Active

Text symbols | Selection texts | List Headings

Name	Text	Dictiona..
P_CLASS	班级	
P_VARI	ALV变式	
P_YEAR	学年	
S_STDNO	学号	

图 2-8

GUI 状态栏编辑如图 2-9 所示（由标准程序 SAPLKKBL 中的 STANDARD 拷贝而成）。

Maintain Status ZTEST_ALV of Interface ZREP003_ALV00_ADD

Function Code Delete subobject

用户界面	ZREP003_ALV00_ADD 不活动的
菜单栏	Standard User Interface STANDARD_FULLSCREEN
应用工具条	STANDARD_FULLSCREEN
功能键	Standard maximale Interaktion STANDARD_FULLSCREEN

图 2-9

2.1.2 程序测试

运行程序结果如图 2-10 所示。

成绩表

ALV变式	
班级	1A
学年	2009
学号	到

图 2-10

填写相关参数，单击按钮，运行结果如图 2-11 所示。

成绩表

检查表格...

成绩表

作成日 Today:2012/07/11 Time:10:25:41
选择基准 班级:1-A 学年:2009

作成者:YIDA TEC

成绩表行标题

班	学...	学...	姓名	Σ 语文成绩	Σ 数学成绩	Σ 自然成绩	Σ 思品成绩
1	2009	00001	郑喜定	60	68	70	81
1A	2009	00002	魏喜奎	62	65	74	83
1A	2009	00003	金喜善	62	65	71	89
1A	2009	00004	洪喜官	61	63	75	88
1A	2009	00005	朴喜珍	64	66	75	88
1A	2009	00006	王喜凤	64	66	75	88
1A	2009	00007	刘喜仁	64	66	76	88
1A	2009	00008	白喜医	62	66	72	88
1A	2009	00009	赵喜庆	62	63	72	82
1A	2009	00010	张喜早	62	63	74	85
				623	**651**	**734**	**860**

图 2-11

2.2 面向对象ALV

2.2.1 Dynpro控件ALV

1. 程序设计

程序源码设计如下：

```
*----------------------------------------------------------------------*
*
* Program ID              ： ZREPO03_ALVOO
* Program Description     ： 成绩查询面向对象 ALV 显示
* Create   User           ： BCUSER
* Create   Date           ： 2011/02/20
*
*----------------------------------------------------------------------*
* Function Overview
*  程序实现以班级，年度，学生为参照进行查询
*    技术点：面向对象 ALV 形式显示，Dynpro 作为容器
*
*----------------------------------------------------------------------*
* Change List             ：
*      change NO.   Change Date    Change User    Change Detail
*      0000001      2011/02/20     BCUSER         程序创建
```

```
*-----------------------------------------------------------------------*

REPORT ZREPO03_ALVOO
   NO STANDARD PAGE HEADING
   LINE-SIZE 120
   LINE-COUNT 90（3）
   MESSAGE-ID ZTESTMSG.

*-----------------------------------------------------------------------*
*-----------------------------------------------------------------------*
INCLUDE：  ZTEST0N,     "常量定义
                ZTEST0F.      "Form 定义

*-----------------------------------------------------------------------*
* TYPE-POOLS
*-----------------------------------------------------------------------*
TYPE-POOLS SLIS.

*-----------------------------------------------------------------------*
* TYPES
*-----------------------------------------------------------------------*
*   成绩表
TYPES：  BEGIN OF WRK_TEST,
             ZZCLASS     LIKE ZTTEST_T-ZZCLASS,          "班级
             ZZYEAR      LIKE   ZTTEST_T-ZZYEAR,          "学年
             ZZSTDNO     LIKE ZTTEST_T-ZZSTDNO,          "学号
             ZZNAME      LIKE ZMNUMBER_T-ZZNAME,        "姓名
             ZZTENKOK    LIKE ZTTEST_T-ZZTENKOK,         "语文
             ZZTENSAN    LIKE ZTTEST_T-ZZTENSAN,         "数学
             ZZTENRIK    LIKE ZTTEST_T-ZZTENRIK,         "自然
             ZZTENSHA    LIKE ZTTEST_T-ZZTENSHA,         "思想品德
             ZZSUM       TYPE P,                          "总成绩
          END    OF WRK_TEST.

TYPES   TY_ITAB_TEST TYPE STANDARD TABLE OF WRK_TEST.

*-----------------------------------------------------------------------*
* TABLES：
*-----------------------------------------------------------------------*
TABLES：  ZTTEST_T,                            "考试成绩表
            ZMNUMBER_T.                        "学生信息表

*-----------------------------------------------------------------------*
* STATICS
*-----------------------------------------------------------------------*
*STATICS：  XX TYPE X VALUE XX.
```

```
*---------------------------------------------------------------------*
* CONSTANTS
*---------------------------------------------------------------------*
CONSTANTS ：
  BEGIN OF AC,
    PROGRAM_NAME     TYPE SY-REPID             VALUE 'ZREPO03_ALVOO',    "程序
*    控制
    STRUCTURE_NAME  TYPE DD02L-TABNAME            VALUE 'ZZTEST_S', "结构体
    ZZCLASS              TYPE DD03L-FIELDNAME  VALUE 'ZZCLASS',         "班级
    ZZYEAR               TYPE DD03L-FIELDNAME  VALUE 'ZZYEAR',          "学年
    ZZSTDNO              TYPE DD03L-FIELDNAME  VALUE 'ZZSTDNO',         "学号
    F2CODE（4）             TYPE C                    VALUE 'POP',             "F2（双击）
    X（1）                  TYPE C                    VALUE 'X',
    H（1）                  TYPE C                    VALUE 'H',
    S（1）                  TYPE C                    VALUE 'S',
    A（1）                  TYPE C                    VALUE 'A',
  END OF AC.

*---------------------------------------------------------------------*
* DATA
*---------------------------------------------------------------------*
DATA：
  BEGIN OF TNAME,
    NAME1 LIKE ZMTNUMBER_T-ZTNAME,                    "班主任
    NAME2 LIKE ZMTNUMBER_T-ZTNAME,                    "副班主任
  END OF TNAME.

DATA：     TBL_TEST    TYPE TY_ITAB_TEST,               "成绩表
       STR_TEST    TYPE WRK_TEST.                       "表头

DATA：   BEGIN OF TBL_TEST1 OCCURS 0.
            INCLUDE STRUCTURE STR_TEST.

DATA：     LIGHT,                                        "例外
           LINECOLOR（4）,                               "颜色
           HL_FNAME    TYPE I,                           "字段
           CELLCOLOR TYPE LVC_T_SCOL,                    "Cell 颜色
           CELLEDIT TYPE LVC_T_STYL,                     "Cell 编辑
           HOTSPOT TYPE STRING.                          "连接
DATA：   END OF TBL_TEST1.
*   ALV 数据
DATA ：
  BEGIN OF AW,
    FIELDCAT       TYPE    SLIS_T_FIELDCAT_ALV ,        "字段属性
    FLDCAT_R       TYPE    SLIS_FIELDCAT_ALV,           "字段的内容
```

```
     LAYOUT        TYPE   SLIS_LAYOUT_ALV,           "布局
     SORT_REC      TYPE   SLIS_SORTINFO_ALV,         "排序记录
     SORTER        TYPE   SLIS_T_SORTINFO_ALV,       "排序属性
     VARIANT       TYPE   DISVARIANT,                "变式
     GRID_SET      TYPE   LVC_S_GLAY,                "表格设定
     EXIT（1）     TYPE   C,
     TITLE         TYPE   LVC_TITLE,                 "成绩表标题
     GRID_TITLE  TYPE   LVC_TITLE,              "成绩表行标题 ',

   END OF AW.
* ALV 编辑设定
DATA：  STR_CELLCOLOR LIKE LINE OF TBL_TEST1-CELLCOLOR,
        STR_CELLEDIT   LIKE LINE OF TBL_TEST1-CELLEDIT.

DATA：  "Tbl_FIELDCAT     TYPE LVC_T_FCAT,            "字段目录
        TBL_HYPETAB      TYPE LVC_T_HYPE.            "超级链接
DATA：  STR_HYPE          TYPE LVC_S_HYPE.            "超级链接

*DATA：  LS_TOOLBAR TYPE STB_BUTTON.
DATA：WCL_CONTAINER TYPE REF TO CL_GUI_CUSTOM_CONTAINER,
       WCL_ALV            TYPE REF TO CL_GUI_ALV_GRID .
*  存放字段目录的内表
DATA TBL_FIELDCAT TYPE LVC_T_FCAT .
DATA：  STR_FIELDCAT     LIKE LINE OF TBL_FIELDCAT.

*  布局结构
DATA：  WK_LAYOUT      TYPE LVC_S_LAYO .
*  预定义事件处理的一个局部类，参考变量声明定义。
CLASS CLS_EVENT_RECEIVER DEFINITION DEFERRED.
DATA：  WO_EVENT_RECEIVER TYPE REF TO CLS_EVENT_RECEIVER.
*  表头信息 TOP OF PAGE
DATA：  W_DOC          TYPE REF TO CL_DD_DOCUMENT,
        W_SPLITTER TYPE REF TO CL_GUI_SPLITTER_CONTAINER,
        W_VIEWER     TYPE REF TO CL_GUI_HTML_VIEWER.

*---------------------------------------------------------------------*
*           CLASS CLS_EVENT_RECEIVER DEFINITION
*---------------------------------------------------------------------*
*           类 CLS_EVENT_RECEIVER 定义
*---------------------------------------------------------------------*
CLASS CLS_EVENT_RECEIVER DEFINITION.

   PUBLIC SECTION.
*     事件方法定义
     METHODS：
        HANDLE_TOOLBAR                                 "工具栏编辑方法
```

```
          FOR EVENT TOOLBAR OF CL_GUI_ALV_GRID
              IMPORTING E_OBJECT E_INTERACTIVE,

      HANDLE_USER_COMMAND                                    "用户事件
          FOR EVENT USER_COMMAND OF CL_GUI_ALV_GRID
              IMPORTING E_UCOMM,

      HANDLE_TOP_OF_PAGE                                   "编辑页头
          FOR EVENT TOP_OF_PAGE    OF CL_GUI_ALV_GRID
              IMPORTING E_DYNDOC_ID TABLE_INDEX,

      HANDLE_DOUBLECLICK                                   "双击事件
          FOR EVENT DOUBLE_CLICK    OF    CL_GUI_ALV_GRID
              IMPORTING E_ROW E_COLUMN ES_ROW_NO.
      ......

      DATA：  CLASS TYPE ZTTEST_T-ZZCLASS.
    PROTECTED SECTION .

ENDCLASS.                    "CLS_EVENT_RECEIVER DEFINITION

*---------------------------------------------------------------------*
*           CLASS CLS_EVENT_RECEIVER IMPLEMENTATION
*---------------------------------------------------------------------*
*           类 CLS_EVENT_RECEIVER 实现
*---------------------------------------------------------------------*
CLASS CLS_EVENT_RECEIVER IMPLEMENTATION.
* 定义工具栏
  METHOD HANDLE_TOOLBAR.
    DATA：    LS_TOOLBAR      TYPE STB_BUTTON.
*     分隔符
    CLEAR    LS_TOOLBAR.
    MOVE     3                    TO LS_TOOLBAR-BUTN_TYPE.
    APPEND LS_TOOLBAR       TO E_OBJECT->MT_TOOLBAR.
*     按钮
    CLEAR LS_TOOLBAR.
    MOVE 'DETAIL'                 TO LS_TOOLBAR-FUNCTION.
    MOVE ICON_EMPLOYEE    TO LS_TOOLBAR-ICON.
    MOVE '个人信息资料' TO LS_TOOLBAR-QUICKINFO.
    MOVE '个人信息'        TO LS_TOOLBAR-TEXT.
    MOVE ' '                          TO LS_TOOLBAR-DISABLED.
    APPEND LS_TOOLBAR       TO E_OBJECT->MT_TOOLBAR.

  ENDMETHOD.                         "handle_toolbar
*
* 定义用户相应
METHOD HANDLE_USER_COMMAND
```

```
*   声明局部变量
    DATA：  LT_ROWS TYPE LVC_T_ROW.
    DATA：  LS_ROWS TYPE LVC_S_ROW.
    DATA：  LV_TXT1 TYPE STRING VALUE '父亲'.
    DATA：  LV_TXT2 TYPE STRING VALUE '母亲'.
    DATA：  LV_TXT3 TYPE STRING VALUE '电话'.
    DATA：  LV_TXT4 TYPE STRING VALUE '地址'.
*   判断操作
    CASE E_UCOMM.
*     点击详细按钮
      WHEN 'DETAIL'.
*       取得ALV 选择行
        CALL METHOD WCL_ALV->GET_SELECTED_ROWS
          IMPORTING
            ET_INDEX_ROWS = LT_ROWS[].
        READ TABLE LT_ROWS INTO LS_ROWS INDEX 1.
*       取得成功
        IF SY-SUBRC = 0.
*         取得对应内表的值
          READ TABLE TBL_TEST INTO STR_TEST INDEX   LS_ROWS-INDEX.
*         取得成功
          IF SY-SUBRC = 0.
*           取得家庭信息
            SELECT   SINGLE    ZZFNAME     "父亲
                               ZZMNAME     "母亲
                               ZZMOBILE    "电话
                               ZZADDR      "地址
              INTO （ZMNUMBER_T-ZZFNAME,
                     ZMNUMBER_T-ZZMNAME,
                     ZMNUMBER_T-ZZMOBILE,
                     ZMNUMBER_T-ZZADDR ）
              FROM   ZMNUMBER_T                          "学生信息表
                   WHERE   ZZNUMBER   = STR_TEST-ZZSTDNO ."学号
*           取得成功
            IF SY-SUBRC EQ 0.
              CONCATENATE  LV_TXT1  ZMNUMBER_T-ZZFNAME    INTO    LV_TXT1
SEPARATED BY SPACE.
              CONCATENATE  LV_TXT2  ZMNUMBER_T-ZZMNAME    INTO    LV_TXT2
SEPARATED BY SPACE.
              CONCATENATE  LV_TXT3  ZMNUMBER_T-ZZMOBILE   INTO    LV_TXT3
SEPARATED BY SPACE.
              CONCATENATE  LV_TXT4  ZMNUMBER_T-ZZADDR     INTO    LV_TXT4
SEPARATED BY SPACE.
              CALL METHOD CL_GUI_CFW=>FLUSH.

*             添加用户响应
```

```
          CALL FUNCTION 'POPUP_TO_INFORM'
            EXPORTING
              TITEL = '学生家庭信息'
              TXT1    = LV_TXT1
              TXT2    = LV_TXT2
              TXT3    = LV_TXT3
              TXT4    = LV_TXT4.

        ENDIF.

      ENDIF.

    ENDIF.
  WHEN OTHERS.

  ENDCASE.
ENDMETHOD.                                "handle_user_command
*
* 编辑表头信息
METHOD HANDLE_TOP_OF_PAGE.
  DATA：  LW_BACK        TYPE SDYDO_KEY VALUE SPACE,
        LW_TEXT        TYPE SDYDO_TEXT_ELEMENT.

  DATA：LV_ZZTEANOM TYPE ZCCLASS_T-ZZTEANOM,          "班主任
       LV_ZZTEANOS TYPE ZCCLASS_T-ZZTEANOS ,          "副班主任
       LV_ZTNAMEP   TYPE ZMTNUMBER_T-ZTNAME ,         "班主任名
       LV_ZTNAMES   TYPE ZMTNUMBER_T-ZTNAME ,         "副班主任名
       LV_CLASS_TEXT TYPE STRING.
*   宏定义
  DEFINE D_TOP_OF_PAGE.
*     设定间隔
    IF &1 IS NOT INITIAL.
      CALL METHOD E_DYNDOC_ID->ADD_GAP
        EXPORTING
          WIDTH = &1.
    ENDIF.
*     新行
    IF &2 IS NOT INITIAL.
      CALL METHOD E_DYNDOC_ID->NEW_LINE.
    ENDIF.
*     添加文本
    IF &3 IS NOT INITIAL.
      LW_TEXT = &3.
      CALL METHOD E_DYNDOC_ID->ADD_TEXT
        EXPORTING
          TEXT          = LW_TEXT
```

```
            SAP_STYLE = &4.
      ENDIF.
    END-OF-DEFINITION.
* 取得班级文本
    SELECT   SINGLE   DD07T～DDTEXT                          "域固定值文本
    INTO   LV_CLASS_TEXT                                     "域固定值文本
    FROM   DD07T INNER JOIN   DD07L                          "内结合
    ON   DD07T～DOMNAME = DD07L～DOMNAME                      "结合条件
    AND DD07T～VALPOS   = DD07L～VALPOS
    WHERE   DD07L～DOMNAME = 'ZZCLASS_D'                     "域名相等
    AND DD07L～DOMVALUE_L = CLASS.                           "域固定值相等
* 取得班主任信息
    SELECT   SINGLE   ZCCLASS_T～ZZTEANOM                    "班主任
                    ZCCLASS_T～ZZTEANOS                      "副班主任
                              A～ZTNAME                      "班主任名
                              B～ZTNAME                      "副班主任名
                 FROM   ZCCLASS_T                            "班主任表
                 INNER JOIN ZMTNUMBER_T   AS A
                 ON   ZCCLASS_T～ZZTEANOM = A～ZTNUMBER
                 INNER JOIN ZMTNUMBER_T   AS B
                 ON ZCCLASS_T～ZZTEANOS = B～ZTNUMBER
                 INTO （LV_ZZTEANOM,LV_ZZTEANOS,LV_ZTNAMEP,LV_ZTNAMES）
                 WHERE   ZZCLASS   = CLASS .                 "班级
*   编辑班级信息
    CONCATENATE TEXT-003 '（'   LV_CLASS_TEXT '）' INTO LV_CLASS_TEXT   .
*   编辑表头
    D_TOP_OF_PAGE：   '50'   ' '   LV_CLASS_TEXT   CL_DD_DOCUMENT=>HEADING,
                      ' '    'X'   ' '                 '',
                      ' '    ' '   '班   主   任'   CL_DD_DOCUMENT=>STRONG,
                      '10'   ' '   LV_ZZTEANOM      CL_DD_DOCUMENT=>KEY,
                  '10' ''   LV_ZTNAMEP        CL_DD_DOCUMENT=>LARGE,
                  ' '    'X' ' '                 '',
                  ' '    ' '  '副班主任'        CL_DD_DOCUMENT=>STRONG,
                  '10'   ' '  LV_ZZTEANOS       CL_DD_DOCUMENT=>KEY,
                  '10'   ' '  LV_ZTNAMES        CL_DD_DOCUMENT=>LARGE,
                  ' '    'X' ' '                 ''.
*                         " T-CODE：   OAER/OAOR
*                         " CLASSNAME：        PICTURES
*                         " CLASSTYPE：        OT
*                         " OBJECT KEY        BOY
*                         " CREATE：           SCREEN
*   添加相片
    CALL METHOD E_DYNDOC_ID->ADD_PICTURE
      EXPORTING
        PICTURE_ID = 'GRADUATION'
        WIDTH        = '100'.
```

```
*     显示表头
      W_DOC->HTML_CONTROL = W_VIEWER.
      CALL METHOD W_DOC->DISPLAY_DOCUMENT
        EXPORTING
          REUSE_CONTROL  = 'X'
          PARENT         = W_HEAD.

  ENDMETHOD.                    "HANDLE_TOP_OF_PAGE
*
* 编辑双击事件
  METHOD   HANDLE_DOUBLECLICK.

      DATA   LV_SEL_COL      TYPE I.                  "列
      DATA：  LV_SEL_ROW      TYPE I,                  "行
            LV_ROWID         TYPE LVC_S_ROW.          "行描述

      DATA：  LV_COL           TYPE STRING,
            LV_ROW           TYPE STRING,
            LV_MSG           TYPE STRING.
*     取得当前行
      CALL METHOD WCL_ALV->GET_CURRENT_CELL
        IMPORTING
          E_ROW       = LV_SEL_ROW
          E_COL       = LV_SEL_COL
          ES_ROW_ID = LV_ROWID.
*     非合计行时
      IF LV_ROWID-ROWTYPE = SPACE.
        LV_COL = LV_SEL_COL.
        LV_ROW   = LV_SEL_ROW.
        CONCATENATE '您双击的是' LV_ROW '行'   LV_COL '列，非合计行。' INTO LV_MSG .
        MESSAGE   LV_MSG     TYPE 'I'.

      ENDIF.

  ENDMETHOD.                    "catch_doubleclick

ENDCLASS.                    "lcl_event_receiver IMPLEMENTATION

*-------------------------------------------------------------------*
* PARAMETERS
*-------------------------------------------------------------------*
PARAMETERS：  P_VARI          TYPE DISVARIANT-VARIANT.      "表示布局（ALV 用）
PARAMETERS：  P_CLASS         TYPE ZTTEST_T-ZZCLASS         "班级
                            OBLIGATORY .                 "必须
PARAMETERS：  P_YEAR TYPE ZTTEST_T-ZZYEAR OBLIGATORY. "学年
```

```
*---------------------------------------------------------------------*
* SELECT-OPTIONS
*---------------------------------------------------------------------*
SELECT-OPTIONS S_STDNO     FOR ZTTEST_T-ZZSTDNO              "学号
                           MATCHCODE OBJECT ZZMCSBID.

*---------------------------------------------------------------------*
* INITIALIZATION
*---------------------------------------------------------------------*
INITIALIZATION.

*  数据初始化
  PERFORM DATA_INIT
            USING
              P_CLASS      "班级
              P_YEAR.      "学年
*---------------------------------------------------------------------*
* AT SELECTION SCREEN
*---------------------------------------------------------------------*
AT SELECTION-SCREEN.

*  选择画面项目检查-班级
  PERFORM CLASS_INPUT_CHECK USING P_CLASS TEXT-003 CHANGING TNAME.

*  选择画面项目检查-学年
  PERFORM YEAR_INPUT_CHECK USING P_YEAR TEXT-004.

*---------------------------------------------------------------------*
* AT SELECTION SCREEN ON VALUE-REQUEST
*---------------------------------------------------------------------*
AT SELECTION-SCREEN ON VALUE-REQUEST FOR   P_VARI .

*  选择画面项目选择-布局
  PERFORM VARIANT_INPUT_SELECT.

*---------------------------------------------------------------------*
*START-OF-SELECTION
*---------------------------------------------------------------------*
START-OF-SELECTION.

*  报表数据初始化
  PERFORM REPORT_DATA_INIT.
*  报表数据选择
  PERFORM REPORT_DATA_SELECT.
*  报表数据编辑
  PERFORM REPORT_DATA_EDIT.
```

```
*  报表数据显示
   CALL SCREEN 9000.

*&---------------------------------------------------------------------*
*&         Form   VARIANT_INPUT_SELECT
*&---------------------------------------------------------------------*
*          选择画面项目选择
*----------------------------------------------------------------------*
FORM VARIANT_INPUT_SELECT .
   AW-VARIANT-REPORT = SY-REPID.          "程序名
   CALL FUNCTION 'REUSE_ALV_VARIANT_F4'
     EXPORTING
       IS_VARIANT          = AW-VARIANT
       I_SAVE              = AC-A
*      it_default_fieldcat    =
     IMPORTING
       E_EXIT              = AW-EXIT
       ES_VARIANT          = AW-VARIANT
     EXCEPTIONS
       NOT_FOUND = 2.
   IF ( SY-SUBRC = 2 ).
     MESSAGE ID SY-MSGID TYPE CNS-MTY_S    NUMBER SY-MSGNO
               WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
   ELSE.
     IF ( AW-EXIT = SPACE ).
       P_VARI = AW-VARIANT-VARIANT.
     ENDIF.
   ENDIF.

ENDFORM.                              " VARIANT_INPUT_SELECT
*&---------------------------------------------------------------------*
*&         Form   REPORT_DATA_INIT
*&---------------------------------------------------------------------*
*          报表数据初始化
*----------------------------------------------------------------------*
FORM REPORT_DATA_INIT .

*  内部表初始化
   CLEAR：      STR_TEST.                          "成绩表头
   REFRESH：   TBL_TEST.                          "成绩表
ENDFORM.                              " REPORT_DATA_INIT
*&---------------------------------------------------------------------*
*&         Form   REPORT_DATA_SELECT
*&---------------------------------------------------------------------*
*          报表数据选择
*----------------------------------------------------------------------*
```

```
FORM REPORT_DATA_SELECT .

* 成绩表数据抽取
  SELECT      ZZCLASS                  "班级
              ZZYEAR                   "学年
              ZZSTDNO                  "学号
              ZZNAME                   "姓名
              ZZTENKOK                 "语文
              ZZTENSAN                 "数学
              ZZTENRIK                 "自然
              ZZTENSHA                 "思品
    FROM ZZTEST_V                      "成绩表
    INTO TABLE TBL_TEST                "成绩内部表
    WHERE ZZCLASS   =   P_CLASS        "班级
    AND     ZZYEAR = P_YEAR            "学年
    AND     ZZSTDNO    IN S_STDNO .    "学号
* 无数据情况
  IF SY-SUBRC <> 0.
    MESSAGE S002.
    STOP.
  ENDIF.
  LOOP AT TBL_TEST INTO STR_TEST.
    STR_TEST-ZZSUM = STR_TEST-ZZSUM +：
                     STR_TEST-ZZTENKOK,
                     STR_TEST-ZZTENSAN,
                     STR_TEST-ZZTENRIK,
                     STR_TEST-ZZTENSHA .
    MODIFY TBL_TEST FROM STR_TEST.


  ENDLOOP.

* 排序
  SORT TBL_TEST BY ZZCLASS ASCENDING          "班级
                   ZZYEAR ASCENDING           "学年
                   ZZSTDNO ASCENDING.         "学号

ENDFORM.                          " REPORT_DATA_SELECT
*&---------------------------------------------------------------------*
*&        Form    REPORT_DATA_EDIT
*&---------------------------------------------------------------------*
*          报表数据编辑
*----------------------------------------------------------------------*
FORM REPORT_DATA_EDIT .
* 准备获取字段目录
  PERFORM PREPARE_FIELD_CATALOG .
```

```
*  设置布局
   PERFORM PREPARE_ALV_LAYOUT .

ENDFORM.                                " REPORT_DATA_EDIT
*&---------------------------------------------------------------------*
*&          Module    INIT_9000    OUTPUT
*&---------------------------------------------------------------------*
*           初期处理
*----------------------------------------------------------------------*
MODULE INIT_9000 OUTPUT.
*  初始化
   IF WCL_ALV IS INITIAL .
*     设置超级链接
      CLEAR：  STR_HYPE, TBL_HYPETAB.
      STR_HYPE-HANDLE = '1'. STR_HYPE-HREF = 'http：//www.51job.com'.
      APPEND STR_HYPE TO TBL_HYPETAB.
      STR_HYPE-HANDLE = '2'. STR_HYPE-HREF = 'http：//www.sap.com'.
      APPEND STR_HYPE TO TBL_HYPETAB.
*     编辑字段信息
      REFRESH：  TBL_TEST1.
      LOOP AT TBL_TEST INTO STR_TEST.
         MOVE-CORRESPONDING STR_TEST TO TBL_TEST1.
*        设定例外信息
         CLEAR STR_CELLCOLOR.
         CLEAR TBL_TEST1-LINECOLOR.
         IF STR_TEST-ZZSUM < 280.
            TBL_TEST1-LIGHT = 1.
            TBL_TEST1-LINECOLOR = 'C611'.
         ELSEIF STR_TEST-ZZSUM >= 280 AND STR_TEST-ZZSUM < 290 .
            TBL_TEST1-LIGHT = 2.
            TBL_TEST1-LINECOLOR = 'C311'.
         ELSE.
            TBL_TEST1-LIGHT = 3.
            TBL_TEST1-LINECOLOR = 'C110'.
         ENDIF.
*        选定第八行
         IF SY-TABIX = 8.
*           所有字段设置为浅红
            STR_CELLCOLOR-COLOR-COL = 6.
            APPEND STR_CELLCOLOR TO TBL_TEST1-CELLCOLOR.
*        选定第九行
         ELSEIF SY-TABIX = 9.
*           语文设置棕色
            STR_CELLCOLOR-FNAME = 'ZZTENKOK'.
            STR_CELLCOLOR-COLOR-COL = 3.
            APPEND STR_CELLCOLOR TO TBL_TEST1-CELLCOLOR.
```

```
*        数学设置绿色
          STR_CELLCOLOR-FNAME = 'ZZTENSAN'.
          STR_CELLCOLOR-COLOR-COL = 5.
          APPEND STR_CELLCOLOR TO TBL_TEST1-CELLCOLOR.
        ENDIF.

        CLEAR STR_CELLEDIT.
*       选定第二行
        IF sy-tabix = 2.
*         设定主页为可编辑
          str_celledit-fieldname = 'HOTSPOT'.
          str_celledit-style = cl_gui_alv_grid=>mc_style_enabled.
          INSERT str_celledit INTO TABLE TBL_TEST1-celledit.

        ENDIF.
*       设置主页内容
        IF SY-TABIX = 1.
          TBL_TEST1-HOTSPOT = 'http：//www.51job.com'.
        ELSE.
          TBL_TEST1-HOTSPOT = 'http：//www.sap.com'.
        ENDIF.
        READ TABLE TBL_HYPETAB INTO STR_HYPE WITH KEY HREF = TBL_TEST1-HOTSPOT.
        IF SY-SUBRC = 0.
          TBL_TEST1-HL_FNAME = STR_HYPE-HANDLE.
        ENDIF.

        APPEND TBL_TEST1.
        REFRESH：  TBL_TEST1-CELLCOLOR.
        REFRESH：  TBL_TEST1-CELLEDIT.

      ENDLOOP.
*     创建控件容器
      CREATE OBJECT：  WCL_CONTAINER
      EXPORTING
      CONTAINER_NAME = 'ALV_CON'.
*     分割大小
      CREATE OBJECT W_SPLITTER
        EXPORTING
          PARENT   = WCL_CONTAINER
          ROWS     = 2
          COLUMNS = 1.
*     将W_HEAD 设定为第一行
      CALL METHOD W_SPLITTER->GET_CONTAINER
        EXPORTING
          ROW              = 1
```

```
        COLUMN          = 1
      RECEIVING
        CONTAINER = W_HEAD.
*     将 W_BODY 设定为第 2 行
      EXPORTING
        ROW             = 2
        COLUMN          = 1
      RECEIVING
        CONTAINER = W_BODY.
*     设定行高
    CALL METHOD W_SPLITTER->SET_ROW_HEIGHT
      EXPORTING
        ID        = 1
        HEIGHT = 30.
*     创建 WCL_ALV 指向 W_BODY
    CREATE OBJECT WCL_ALV    EXPORTING I_PARENT = W_BODY.
  ENDIF.

  AW-VARIANT-REPORT       = SY-REPID.
  WK_LAYOUT-GRID_TITLE = TEXT-001.
* 允许多行选择
  WK_LAYOUT-SEL_MODE      = 'A'.
  WK_LAYOUT-EXCP_FNAME = 'LIGHT'.
  WK_LAYOUT-INFO_FNAME = 'LINECOLOR'.
  WK_LAYOUT-CTAB_FNAME = 'CELLCOLOR'.
  WK_LAYOUT-STYLEFNAME = 'CELLEDIT'.

* 初始化 WCL_ALV

  CALL METHOD WCL_ALV->REGISTER_EDIT_EVENT
    EXPORTING
      I_EVENT_ID = CL_GUI_ALV_GRID=>MC_EVT_ENTER.

  CALL METHOD WCL_ALV->SET_READY_FOR_INPUT
    EXPORTING
      I_READY_FOR_INPUT = 1.
* 显示 ALV
  CALL METHOD WCL_ALV->SET_TABLE_FOR_FIRST_DISPLAY
  EXPORTING
*       I_BUFFER_ACTIVE =                        " 激活缓存
```

```
*       I_CONSISTENCY_CHECK =
*       I_STRUCTURE_NAME =                          " DDIC 结构体
     IS_VARIANT =    AW-VARIANT                     "  变式
      I_SAVE = 'A'                                  "  变式保存制限
      I_DEFAULT = 'X'                               "  变式初期设定
     IS_LAYOUT = WK_LAYOUT                          "  布局样式
*       IS_PRINT =                                  "  印刷
*       IT_SPECIAL_GROUPS =                         "  分组
*       IT_TOOLBAR_EXCLUDING =                      "  隐藏功能菜单项
      IT_HYPERLINK =          TBL_HYPETAB           "  设置超级链接
   CHANGING
     IT_OUTTAB = TBL_TEST1[]
     IT_FIELDCATALOG = TBL_FIELDCAT
*       IT_SORT =                               "  设置排序
*       IT_FILTER =                             "  设置筛选
   EXCEPTIONS
     INVALID_PARAMETER_COMBINATION = 1
     PROGRAM_ERROR = 2
     TOO_MANY_LINES = 3
     OTHERS = 4 .

* 创建 ALV 控件
   IF WO_EVENT_RECEIVER IS INITIAL.
     CREATE OBJECT WO_EVENT_RECEIVER.
   ENDIF.
   WO_EVENT_RECEIVER->CLASS = P_CLASS.
* 注册 ALV 各事件的处理程序
   SET HANDLER WO_EVENT_RECEIVER->HANDLE_USER_COMMAND FOR WCL_ALV .
   SET HANDLER WO_EVENT_RECEIVER->HANDLE_TOOLBAR         FOR WCL_ALV .
   SET HANDLER WO_EVENT_RECEIVER->HANDLE_TOP_OF_PAGE    FOR WCL_ALV .
   SET HANDLER WO_EVENT_RECEIVER->HANDLE_DOUBLECLICK    FOR WCL_ALV .
* 触发 TOP_OF_PAGE 事件
   IF W_DOC IS INITIAL.
     CREATE OBJECT W_DOC.
   ENDIF.
   CALL METHOD WCL_ALV->LIST_PROCESSING_EVENTS
     EXPORTING
       I_EVENT_NAME = 'TOP_OF_PAGE'
       I_DYNDOC_ID    = W_DOC.
```

```
* 激活工具栏
  CALL METHOD WCL_ALV->SET_TOOLBAR_INTERACTIVE.

* 设置光标在 ALV 上
  CALL METHOD CL_GUI_CONTROL=>SET_FOCUS
    EXPORTING
      CONTROL = WCL_ALV.
* 刷新 ALV
  CALL METHOD WCL_ALV->REFRESH_TABLE_DISPLAY.

ENDMODULE.                    " INIT_9000  OUTPUT
*&---------------------------------------------------------------------*
*&      Form  PREPARE_FIELD_CATALOG
*&---------------------------------------------------------------------*
*       准备获取字段目录
*----------------------------------------------------------------------*
FORM PREPARE_FIELD_CATALOG .
  DATA LS_FCAT TYPE LVC_S_FCAT .
* 通过 DDIC 结构体获取字段目录
  CALL FUNCTION 'LVC_FIELDCATALOG_MERGE'
    EXPORTING
      I_STRUCTURE_NAME         = AC-STRUCTURE_NAME"DDIC 结构体
    CHANGING
      CT_FIELDCAT              = TBL_FIELDCAT[]
    EXCEPTIONS
      INCONSISTENT_INTERFACE   = 1
      PROGRAM_ERROR            = 2
      OTHERS                   = 3.
* 异常处理
  IF SY-SUBRC <> 0.

    MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
            WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ENDIF.
* 修改字段目录
  LOOP AT TBL_FIELDCAT INTO LS_FCAT .
    CASE LS_FCAT-FIELDNAME .
      WHEN 'ZZNAME' .
        LS_FCAT-OUTPUTLEN = '10' .
```

```
          MODIFY TBL_FIELDCAT FROM LS_FCAT .
        WHEN 'ZZCLASS' OR 'ZZYEAR' .
          LS_FCAT-NO_OUT = 'X' .
          MODIFY TBL_FIELDCAT FROM LS_FCAT .

    ENDCASE .
  ENDLOOP .
* 总成绩
  LS_FCAT-FIELDNAME = 'ZZSUM'.
  LS_FCAT-COLTEXT = '总成绩' .
  APPEND   LS_FCAT TO TBL_FIELDCAT .
  CLEAR LS_FCAT.
* 超级链接
  LS_FCAT-COL_POS    = LINES( TBL_FIELDCAT ) + 1.
  LS_FCAT-FIELDNAME    = 'HOTSPOT'.
  LS_FCAT-REPTEXT = '个人主页'.
  LS_FCAT-OUTPUTLEN = '20'.
  LS_FCAT-WEB_FIELD = 'HL_FNAME'.
  APPEND LS_FCAT TO TBL_FIELDCAT.


ENDFORM.                              " PREPARE_FIELD_CATALOG
*&---------------------------------------------------------------------*
*&         Form   PREPARE_ALV_LAYOUT
*&---------------------------------------------------------------------*
*          设置布局
*----------------------------------------------------------------------*
FORM PREPARE_ALV_LAYOUT .
  WK_LAYOUT-ZEBRA = 'X' .                         "可选行颜色  （带）
  WK_LAYOUT-GRID_TITLE = TEXT-002 .        "标题
  WK_LAYOUT-SMALLTITLE = 'X' .                 "标题大小

ENDFORM.                              " PREPARE_ALV_LAYOUT
*&---------------------------------------------------------------------*
*&         Module   STATUS_9000   OUTPUT
*&---------------------------------------------------------------------*
*          STATUS 设定处理
*----------------------------------------------------------------------*
MODULE STATUS_9000 OUTPUT.
```

```
    SET PF-STATUS 'ZTEST_ALVOO'.
    SET TITLEBAR 'ALVOO'.

ENDMODULE.                    " STATUS_9000    OUTPUT
*&---------------------------------------------------------------------*
*&          Module    EXIT_COMMAND_9000    INPUT
*&---------------------------------------------------------------------*
*           程序终了处理
*----------------------------------------------------------------------*
MODULE EXIT_COMMAND_9000 INPUT.

    LEAVE TO SCREEN 0.

ENDMODULE.                    " EXIT_COMMAND_9000    INPUT
```

GUI 状态栏编辑如图 2-12 所示（由标准程序 SAPLKKBL 中的 STANDARD 拷贝而成）。

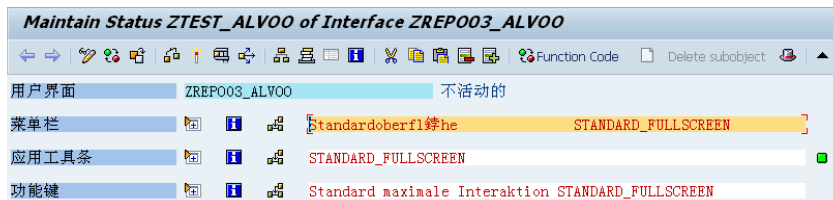


图 2-12

GUI 标题编辑如图 2-13 所示。

用户界面 ZREP003_ALVOO 已激活

标题数量	标题
ALVOO	成绩表ALVOO显示

图 2-13

屏幕 9000，一般属性设定如图 2-14 所示。
元素清单生成如图 2-15 所示。
控制逻辑编辑如下：

```
************************************************************************
* PROCESS BEFORE OUTPUT
************************************************************************
PROCESS BEFORE OUTPUT.

* STATUS 设定处理
    MODULE STATUS_9000.
```

屏幕号码 9000 激活

属性 | 元素清单 | 逻辑流

Short description 成绩表OOALV画面

Original language ZH 中文 开发类 ZABAP_TRAINING

Last changed on/by 11.06.2015 14:18:02

Last generation 11.06.2015 14:30:14

Screen type

◉通常
○子屏幕
○方式对话框
○选择屏幕

Settings

☐暂存数据
☐Switch off runtime compress
☐Template - non-executable
☐Hold Scroll Position
☐Without Application Toolbar

Other attributes

Next Screen 9000

Cursor position

Screen group

Lines/Columns Occupied 26 101

Mainten. 27 120

Context menu FORM ON CTMENU

图 2-14

屏幕号码 9000 激活

属性 | 元素清单 | 逻辑流

General attr. | Texts/ I/O templates | Special attr. | Display attr. | Mod. groups / functions | References

层.	E	名称	屏幕...	行	列	D...	可...	高度	可...	格式	输入	输出	只...	Di...	词...	Property list
		%#AUTOTEXT001	文本	2	39	18	18	1			☐	☐	☐	☐		Properties
		%#AUTOTEXT002	框	3	4	98	98	4			☐	☐	☐	☐		Properties
		%#AUTOTEXT003	文本	4	8	4	4	1			☐	☐	☐	☐		Properties
		P_CLASS	I/O	4	15	2	2	1	☐	CHAR	☐	☑	☐	☐		Properties
		%#AUTOTEXT004	文本	4	27	4	4	1			☐	☐	☐	☐		Properties
		P_YEAR	I/O	4	36	4	4	1	☐	NUMC	☐	☑	☐	☐		Properties
		%#AUTOTEXT005	文本	5	8	4	4	1			☐	☐	☐	☐		Properties
		S_STDNO-LOW	I/O	5	15	5	5	1	☐	CHAR	☐	☑	☐	☐		Properties
		%#AUTOTEXT006	文本	5	27	2	2	1			☐	☐	☐	☐		Properties
		S_STDNO-HIGH	I/O	5	36	5	5	1	☐	CHAR	☐	☑	☐	☐		Properties
+		ALV_CON	CCtrl	7	5	96	96	20			☐	☐	☐	☐		
		OK_CODE	OK	0	0	20	20	1	☐	OK				☐		

图 2-15

```
* 初期处理
  MODULE INIT_9000.

************************************************************************
* PROCESS AFTER INPUT
************************************************************************
PROCESS AFTER INPUT.

* 程序终了处理
  MODULE EXIT_COMMAND_9000 AT EXIT-COMMAND .
```

* *机能处理*

MODULE USER_COMMAND_9000.

画面格式编辑如图 2−16 所示。



图 2−16

注：在生成 CL_GUI_ALV_GRID 实例时，构造器参数 I_PARENT 是必需的，但如果要对 ALV 的功能进行扩展，例如工具栏加入按钮，并在事件处理器中对 ALV 的单元表格的属性如颜色、样式等进行更改，则必须对 I_APPL_EVENTS 赋值为 X。

SET_TABLE_FOR_FIRST_DISPLAY 的一些参数：

隐藏不需要的功能：IT_TOOLBAR_EXCLUDING，相关的功能码是以 MC_FC 打头的常量。

定义排序表：IT_SORT 排序时会把相同的列值垂直合并，如果要对所有列取消合并可更改布局中的 no_merging，更改单列的合并可更改对应字段目录的 no_merging，它主要是通过一个类 CL_SALV_TABLE 来实现的。

字段目录（Field Catalog）：字段目录可以手动生成，也可以调用函数 LVC_FIELDCAT-ALOG_MERGE 先生成一个字段目录表，然后再按需求循环修改属性。I_STRUCTURE_NAME 和 IT_FIELDCATALOG 两个参数如果都传入的话，I_STRUCTURE_NAME 有优先权。

2．程序测试

运行程序结果如图 2−17 所示。

填写相关参数，单击按钮，运行结果如图 2−18 所示。

双击事件测试结果如图 2−19 所示。

自定义功能测试如图 2−20 所示。

成绩表

ALV变式

班级 1A

学年 2009

学号 到

图 2-17

成绩表ALVOO显示

******成绩表******

选择基准

班级 1A 学年 2009

学号 ～

班级(一年级A班)

班 主 任 00001 刘备

副班主任 00002 严长寿

个人信息

成绩表

例外	学生...	姓名	语...	数...	自...	思...	总...	个人主页
	00001	郑喜定	60	68	70	81	279	http://www.51job.com
	00002	魏喜奎	62	65	74	83	284	http://www.sap.com
	00003	金喜善	62	65	71	89	287	http://www.sap.com
	00004	洪喜官	61	63	75	88	287	http://www.sap.com
	00005	朴喜珍	64	66	75	88	293	http://www.sap.com
	00006	王喜凤	64	66	75	88	293	http://www.sap.com
	00007	刘喜仁	64	66	76	88	294	http://www.sap.com
	00008	白喜医	62	66	72	88	288	http://www.51job.com
	00009	赵喜庆	62	63	72	82	279	http://www.sap.com
	00010	张喜早	62	63	74	85	284	http://www.sap.com

图 2-18

信息

您双击的是3 行3 列，非合计行。

图 2-19

学生家庭信息

父亲 洪正哲

母亲 萧芳英英

电话 13987645680

地址 北京鲁迅路131号

图 2-20

2.3 面向对象ALV

2.3.1 程序设计

程序源码编辑如下：

```
*--------------------------------------------------------------------*
*
* Program ID              ：ZREPO03_ALV
* Program Description     ：成绩查询ALV 显示
* Create    User          ：BCUSER
* Create    Date          ：2011/02/20
*
*--------------------------------------------------------------------*
* Function Overview
*  程序实现以班级、年度、学生为参照进行查询
*    技术点：OO-ALV 形式显示，试图应用
*
*--------------------------------------------------------------------*
* Change List          ：
*       change NO.   Change Date    Change User    Change Detail
*       0000001      2011/02/20     BCUSER         程序创建
*--------------------------------------------------------------------*

REPORT ZREPO03_ALV
   NO STANDARD PAGE HEADING
   LINE-SIZE 120
   LINE-COUNT 90(3)
   MESSAGE-ID ZTESTMSG.

*--------------------------------------------------------------------*
* INCLUDE
*--------------------------------------------------------------------*
INCLUDE：   ZTEST0N,      "常量定义
              ZTEST0F.          "Form 定义

*--------------------------------------------------------------------*
* TYPE-POOLS
*--------------------------------------------------------------------*
TYPE-POOLS SLIS.

*--------------------------------------------------------------------*
* TYPES
```

```
*---------------------------------------------------------------------*
*   成绩表
TYPES:   BEGIN OF     TYP_TEST,
           ZZCLASS      LIKE ZTTEST_T-ZZCLASS,          "班级
           ZZYEAR       LIKE   ZTTEST_T-ZZYEAR,         "学年
           ZZSTDNO      LIKE ZTTEST_T-ZZSTDNO,          "学号
           ZZNAME       LIKE ZMNUMBER_T-ZZNAME,         "姓名
           ZZTENKOK     LIKE ZTTEST_T-ZZTENKOK,         "语文
           ZZTENSAN     LIKE ZTTEST_T-ZZTENSAN,         "数学
           ZZTENRIK     LIKE ZTTEST_T-ZZTENRIK,         "自然
           ZZTENSHA     LIKE ZTTEST_T-ZZTENSHA,         "思想品德
         END     OF TYP_TEST.

"TYPES   TYP_ITAB_TEST TYPE STANDARD TABLE OF TYP_TEST .
TYPES   TYP_ITAB_TEST TYPE   TYP_TEST OCCURS 0.

*---------------------------------------------------------------------*
* TABLES:
*---------------------------------------------------------------------*
TABLES:   ZTTEST_T,                             "考试成绩表
          ZMNUMBER_T.                           "学生信息表

*---------------------------------------------------------------------*
* STATICS
*---------------------------------------------------------------------*
*STATICS:   XX TYPE X VALUE XX.

*---------------------------------------------------------------------*
* CONSTANTS
*---------------------------------------------------------------------*
CONSTANTS  :
  BEGIN OF AC,
    PROGRAM_NAME     TYPE SY-REPID          VALUE 'ZREPO03_ALV',     "程序
*   控制
    STRUCTURE_NAME TYPE DD02L-TABNAME   VALUE 'ZZTEST_S',     "结构体
    ZZCLASS          TYPE DD03L-FIELDNAME    VALUE 'ZZCLASS',      "班级
    ZZYEAR           TYPE DD03L-FIELDNAME    VALUE 'ZZYEAR',       "学年
    ZZSTDNO          TYPE DD03L-FIELDNAME    VALUE 'ZZSTDNO',      "学号
    F2CODE(4)        TYPE C                  VALUE 'POP',          "F2（双击）
    X(1)             TYPE C                  VALUE 'X',
    H(1)             TYPE C                  VALUE 'H',
    S(1)             TYPE C                  VALUE 'S',
    A(1)             TYPE C                  VALUE 'A',
```

```
  END OF AC.

*--------------------------------------------------------------------*
* DATA
*--------------------------------------------------------------------*
*  表头信息定义
DATA：
  BEGIN OF TNAME,
    NAME1 LIKE ZMTNUMBER_T-ZTNAME,                    "班主任
    NAME2 LIKE ZMTNUMBER_T-ZTNAME,                    "副班主任
  END OF TNAME.

DATA：    TBL_TEST    TYPE TYP_ITAB_TEST,               "成绩表
              STR_TEST    TYPE TYP_TEST.                "表头
* ALV 信息定义
DATA：  BEGIN OF AW,
        FIELDCAT      TYPE   SLIS_T_FIELDCAT_ALV ,      "字段属性
        FLDCAT_R      TYPE   SLIS_FIELDCAT_ALV,         "字段的内容
        LAYOUT        TYPE   SLIS_LAYOUT_ALV,           "布局
        SORT_REC      TYPE   SLIS_SORTINFO_ALV,         "排序记录
        SORTER        TYPE   SLIS_T_SORTINFO_ALV,       "排序属性
        VARIANT       TYPE   DISVARIANT,                "变式
        GRID_SET      TYPE   LVC_S_GLAY,                "表格设定
        TITLE         TYPE   LVC_TITLE,                 "成绩表标题
        GRID_TITLE    TYPE   LVC_TITLE,                 "成绩表行标题
        EXIT(1)       TYPE   C,
        END OF AW.
DATA：  W_DOC TYPE REF TO CL_DD_DOCUMENT,
        W_SPLITTER TYPE REF TO CL_GUI_SPLITTER_CONTAINER,
        W_HEAD        TYPE REF TO CL_GUI_CONTAINER,
        W_BODY        TYPE REF TO CL_GUI_CONTAINER,
        W_VIEWER      TYPE REF TO CL_GUI_HTML_VIEWER.

*  预定义事件处理的一个局部类，参考变量声明定义
CLASS：  CLS_EVENT_RECEIVER DEFINITION DEFERRED.
DATA：    WO_EVENT_RECEIVER TYPE REF TO CLS_EVENT_RECEIVER.
CLASS：  CLS_ALV_OPERATION DEFINITION DEFERRED.
DATA：    WO_ALV_OPERATION    TYPE REF TO CLS_ALV_OPERATION.
*--------------------------------------------------------------------*
*          CLASS CLS_EVENT_RECEIVER DEFINITION
*--------------------------------------------------------------------*
*          事件处理定义
*--------------------------------------------------------------------*
```

```
CLASS CLS_EVENT_RECEIVER    DEFINITION.

  PUBLIC SECTION.
*     事件方法定义
    CLASS-METHODS：
                    ON_USER_COMMAND           FOR EVENT          "用户事件
      ADDED_FUNCTION        OF     CL_SALV_EVENTS_TABLE
                         IMPORTING E_SALV_FUNCTION,
                    ON_DOUBLE_CLICK           FOR EVENT          "双击事件
      DOUBLE_CLICK              OF     CL_SALV_EVENTS_TABLE
                         IMPORTING ROW COLUMN ,
                    ON_LINK_CLICK              FOR EVENT          "热点单击事件
      LINK_CLICK                 OF     CL_SALV_EVENTS_TABLE
                         IMPORTING ROW COLUMN,
                    ON_BEFORE_SALV_FUNCTION FOR EVENT
      BEFORE_SALV_FUNCTION OF    CL_SALV_EVENTS_TABLE        "响应之前事件
                         IMPORTING E_SALV_FUNCTION,
                    ON_AFTER_SALV_FUNCTION FOR EVENT
      AFTER_SALV_FUNCTION    OF    CL_SALV_EVENTS_TABLE        "响应之后事件
                         IMPORTING E_SALV_FUNCTION.
    ......

  PROTECTED SECTION .
*     定义属性
    CLASS-DATA：    W_MSG_STRING TYPE STRING.                  "信息
    DATA：    CLASS      TYPE    ZTTEST_T-ZZCLASS ,              "班级
            YEAR        TYPE    ZTTEST_T-ZZYEAR   ,             "学年
            STDNO_H TYPE    ZTTEST_T-ZZSTDNO.                 "学号
ENDCLASS.                        "CLS_EVENT_RECEIVER DEFINITION

*-------------------------------------------------------------------*
*           CLASS CLS_EVENT_RECEIVER IMPLEMENTATION
*-------------------------------------------------------------------*
*           事件处理实现
*-------------------------------------------------------------------*
CLASS CLS_EVENT_RECEIVER    IMPLEMENTATION.
*
* 单击功能按钮事件处理方法
  METHOD ON_USER_COMMAND.
    CONCATENATE '按钮事件功能代码为  '
                  E_SALV_FUNCTION
              INTO W_MSG_STRING SEPARATED BY SPACE.
    MESSAGE I000(0K) WITH W_MSG_STRING.
```

```
  ENDMETHOD.                      "on_user_command
*
* 双击事件处理方法
  METHOD ON_DOUBLE_CLICK.
    W_MSG_STRING = ROW.
    CONCATENATE '双击事件在'
                '行'
                W_MSG_STRING
                '列'
                COLUMN
                '上'
          INTO W_MSG_STRING SEPARATED BY SPACE.
    MESSAGE I000（0K）WITH W_MSG_STRING.
  ENDMETHOD.                      "on_double_click
*
* 单击事件处理方法
  METHOD ON_LINK_CLICK.
    DATA：  LV_ZZFNAME TYPE    ZMNUMBER_T-ZZFNAME ,
           LV_ZZMNAME TYPE    ZMNUMBER_T-ZZMNAME .
*     取得当前信息
    READ TABLE TBL_TEST INTO STR_TEST INDEX ROW.
    SELECT     SINGLE       ZZFNAME ZZMNAME                    "父母
               INTO（LV_ZZFNAME,LV_ZZMNAME）
               FROM    ZMNUMBER_T                              "学生信息表
               WHERE    ZZNUMBER    = STR_TEST-ZZSTDNO    .    "学号
*     成功的场合
    IF SY-SUBRC = 0.

      CONCATENATE STR_TEST-ZZNAME
                  '家庭信息是'
                  '父亲：' LV_ZZFNAME
                  '母亲：' LV_ZZMNAME
                  INTO W_MSG_STRING .
      MESSAGE I000（0K）WITH W_MSG_STRING.
    ENDIF.

  ENDMETHOD.                      "on_link_click
*
* 触发事件之前
  METHOD ON_BEFORE_SALV_FUNCTION.
    CONCATENATE '事件  '
                E_SALV_FUNCTION    '触发之前.'
```

```
    MESSAGE I000（0K）WITH W_MSG_STRING.
  ENDMETHOD.                    "on_before_salv_function
*
* 触发事件之后
  METHOD ON_AFTER_SALV_FUNCTION.
    CONCATENATE '事件  '
                E_SALV_FUNCTION '触发之后.'
           INTO W_MSG_STRING SEPARATED BY SPACE.
    MESSAGE I000（0K）WITH W_MSG_STRING.
  ENDMETHOD.                    "on_after_salv_function

ENDCLASS.                    "cls_event_receiver IMPLEMENTATION
*---------------------------------------------------------------------*
*       CLASS CLS_ALV_OPERATIO DEFINITION
*---------------------------------------------------------------------*
*       ALV 操作类（定义）
*---------------------------------------------------------------------*
CLASS CLS_ALV_OPERATION DEFINITION.
  PUBLIC SECTION.
*   定义属性
    DATA： CLASS    TYPE   ZTTEST_T-ZZCLASS ,          "班级
         YEAR      TYPE   ZTTEST_T-ZZYEAR,            "学年
         STDNO_L TYPE   ZTTEST_T-ZZSTDNO,          "学号
         STDNO_H TYPE   ZTTEST_T-ZZSTDNO.          "学号
*   取得显示的成绩数据
    METHODS：  GET_TEST_DATA
               RETURNING VALUE（LT_TEST）TYPE TYP_ITAB_TEST,
*   全屏 Grid 列表处理方法
             SET_ALV_FULL
               IMPORTING VALUE（LT_TEST）TYPE TYP_ITAB_TEST.
  PRIVATE SECTION.
*   定义私有属性
    DATA：  WO_TABLE TYPE REF TO CL_SALV_TABLE.

ENDCLASS.                    "CLS_ALV_OPERATIO DEFINITION

*---------------------------------------------------------------------*
*       CLASS CLS_ALV_OPERATION IMPLEMENTATION
*---------------------------------------------------------------------*
*       ALV 操作类（实现）
*---------------------------------------------------------------------*
CLASS CLS_ALV_OPERATION   IMPLEMENTATION.
*
```

```
* 取得要显示的数据
  METHOD GET_TEST_DATA.
*
*     成绩表数据抽取
      SELECT        ZZCLASS                "班级
                    ZZYEAR                 "学年
                    ZZSTDNO                "学号
                    ZZNAME                 "姓名
                    ZZTENKOK               "语文
                    ZZTENSAN               "数学
                    ZZTENRIK               "自然
                    ZZTENSHA               "思品
        FROM ZZTEST_V                      "成绩表
        INTO TABLE LT_TEST                 "成绩内部表
        WHERE ZZCLASS   =   CLASS          "班级
        AND     ZZSTDNO   BETWEEN STDNO_L AND STDNO_H. "学号
*     无数据情况
      IF SY-SUBRC <> 0.
        MESSAGE S002.
      ENDIF.

*     排序
      SORT TBL_TEST BY  ZZCLASS ASCENDING        "班级
                        ZZYEAR ASCENDING         "学年
                        ZZSTDNO ASCENDING.       "学号

  ENDMETHOD.                        "GET_TEST_DATA
*
* 输出全屏网格列表的方法
  METHOD SET_ALV_FULL.
      DATA：  LR_FUNCTIONS   TYPE REF TO CL_SALV_FUNCTIONS_LIST,
            LR_EVENTS        TYPE REF TO CL_SALV_EVENTS_TABLE,
            LR_COLUMNS       TYPE REF TO CL_SALV_COLUMNS_TABLE,
            LR_COLUMN        TYPE REF TO CL_SALV_COLUMN_TABLE,
            LR_SORTS         TYPE REF TO CL_SALV_SORTS,
            LR_CONTENT       TYPE REF TO CL_SALV_FORM_ELEMENT.

*     创建实例
      TRY.
          CL_SALV_TABLE=>FACTORY(
            IMPORTING
              R_SALV_TABLE = WO_TABLE
            CHANGING
```

```
            T_TABLE         = LT_TEST
        ).
      CATCH CX_SALV_MSG.
    ENDTRY.
*   显示 ALV 标准功能
    LR_FUNCTIONS = WO_TABLE->GET_FUNCTIONS( ).
    LR_FUNCTIONS->SET_ALL('X').
*   列值自动设置
    LR_COLUMNS = WO_TABLE->GET_COLUMNS( ).
    LR_COLUMNS->SET_OPTIMIZE( 'X' ).
*   排序
    LR_SORTS        = WO_TABLE->GET_SORTS( ).
*   创建排序
    TRY .
        LR_SORTS->ADD_SORT(
          COLUMNNAME = 'ZZCLASS'                        "班级
          SEQUENCE     = IF_SALV_C_SORT=>SORT_UP        "升序
        ).
        LR_SORTS->ADD_SORT(
          COLUMNNAME = 'ZZYEAR'                         "学年
          SEQUENCE     = IF_SALV_C_SORT=>SORT_UP        "升序
              ).

      CATCH CX_SALV_DATA_ERROR.
      CATCH CX_SALV_EXISTING.
      CATCH CX_SALV_NOT_FOUND.
    ENDTRY.

*   设置 GUI Status
    WO_TABLE->SET_SCREEN_STATUS(
      SET_FUNCTIONS = WO_TABLE->C_FUNCTIONS_ALL
    ).
*   设置热点
    LR_COLUMNS = WO_TABLE->GET_COLUMNS( ).
    TRY.
        LR_COLUMN ?= LR_COLUMNS->GET_COLUMN( 'ZZSTDNO' ).
        LR_COLUMN->SET_CELL_TYPE( IF_SALV_C_CELL_TYPE=>HOTSPOT ).
        LR_COLUMN->SET_LONG_TEXT( '学号' ).
        LR_COLUMN ?= LR_COLUMNS->GET_COLUMN( 'ZZNAME' ).
        LR_COLUMN->SET_CELL_TYPE( IF_SALV_C_CELL_TYPE=>HOTSPOT ).
        LR_COLUMN->SET_LONG_TEXT( '学年' ).

      CATCH CX_SALV_NOT_FOUND.
```

```
      ENDTRY.
*     注册事件
      LR_EVENTS = WO_TABLE->GET_EVENT( ).
      IF WO_EVENT_RECEIVER IS INITIAL.
        CREATE OBJECT WO_EVENT_RECEIVER.
      ENDIF.
      IF W_DOC IS INITIAL.
        CREATE OBJECT W_DOC.
      ENDIF.

      SET HANDLER CLS_EVENT_RECEIVER=>ON_USER_COMMAND FOR LR_EVENTS.
      SET HANDLER CLS_EVENT_RECEIVER=>ON_DOUBLE_CLICK FOR LR_EVENTS.
      SET HANDLER CLS_EVENT_RECEIVER=>ON_LINK_CLICK FOR LR_EVENTS.
      SET HANDLER CLS_EVENT_RECEIVER=>ON_BEFORE_SALV_FUNCTION FOR LR_
EVENTS.
      SET HANDLER CLS_EVENT_RECEIVER=>ON_AFTER_SALV_FUNCTION FOR LR_
EVENTS.
*     编辑表头
      DATA：  LO_HEADER      TYPE REF TO CL_SALV_FORM_LAYOUT_GRID,
              LO_H_LABEL     TYPE REF TO CL_SALV_FORM_LABEL,
              LO_H_FLOW      TYPE REF TO CL_SALV_FORM_LAYOUT_FLOW.
      DATA：  LISTHEADER     TYPE SLIS_T_LISTHEADER,
              WA_LISTHEADER TYPE SLIS_LISTHEADER,
              H_DATUM(10),   H_UZEIT(10),
              EMPTY(10)      TYPE C VALUE '          '.

*     创建表头对象
      CREATE OBJECT LO_HEADER.

*     标题
      LO_H_LABEL = LO_HEADER->CREATE_LABEL( ROW = 1 COLUMN = 1 ).
      WA_LISTHEADER-INFO     = AW-TITLE.
      IF ( NOT AW-VARIANT-TEXT IS INITIAL ).
        CONCATENATE AW-TITLE '(' AW-VARIANT-TEXT ')'
                    INTO WA_LISTHEADER-INFO.
      ENDIF.
      LO_H_LABEL->SET_TEXT( WA_LISTHEADER-INFO ).
*     中标题
      LO_H_LABEL = LO_HEADER->CREATE_LABEL( ROW = 3 COLUMN = 1 ).
      WRITE SY-DATUM TO H_DATUM DD/MM/YYYY.
      WRITE SY-UZEIT TO H_UZEIT USING EDIT MASK '__：__：__'.
      CLEAR WA_LISTHEADER.
      WA_LISTHEADER-KEY      =   '作成日'.
```

```
    LO_H_LABEL->SET_TEXT( WA_LISTHEADER-KEY    ).

    LO_H_FLOW = LO_HEADER->CREATE_FLOW( ROW = 3 COLUMN = 2 ).
    CONCATENATE 'Today：' H_DATUM    EMPTY
    LO_H_LABEL = LO_HEADER->CREATE_LABEL( ROW = 4 COLUMN = 1 ).
    WA_LISTHEADER-KEY      =   '选择基准'.
    LO_H_LABEL->SET_TEXT( WA_LISTHEADER-KEY ).
    LO_H_FLOW = LO_HEADER->CREATE_FLOW( ROW = 4 COLUMN = 2 ).
    CONCATENATE '班级:' CLASS+0(1) CNS-PLUS     CLASS+1(1)    EMPTY EMPTY
                    '学年：' YEAR
           INTO WA_LISTHEADER-INFO .
    LO_H_FLOW->CREATE_TEXT( TEXT = WA_LISTHEADER-INFO ).
*     小标题
    LO_H_FLOW = LO_HEADER->CREATE_FLOW( ROW = 5 COLUMN = 1 ).
    CLEAR WA_LISTHEADER.
    CONCATENATE '作成者：'    SY-UNAME    INTO WA_LISTHEADER-INFO .
    LO_H_FLOW->CREATE_TEXT( TEXT = '作成者:' ).
    LO_H_FLOW = LO_HEADER->CREATE_FLOW( ROW = 5 COLUMN = 2 ).
    LO_H_FLOW->CREATE_TEXT( TEXT = SY-UNAME    ).
*     设定表头
    WO_TABLE->SET_TOP_OF_LIST( LO_HEADER ).
*     显示表头
    WO_TABLE->SET_TOP_OF_LIST_PRINT( LO_HEADER ).
*     显示列表
    WO_TABLE->DISPLAY( ).
  ENDMETHOD.                        "SET_ALV_FULL
ENDCLASS.                           "CLS_ALV_OPERATION IMPLEMENTATION

*---------------------------------------------------------------------*
* PARAMETERS
*---------------------------------------------------------------------*
PARAMETERS：   P_VARI         TYPE DISVARIANT-VARIANT.       "表示布局（ALV 用）
PARAMETERS   P_CLASS        TYPE ZTTEST_T-ZZCLASS          "班级
                            OBLIGATORY .                   "必须

PARAMETERS P_YEAR TYPE ZTTEST_T-ZZYEAR OBLIGATORY.         "学年

*---------------------------------------------------------------------*
* SELECT-OPTIONS
*---------------------------------------------------------------------*
SELECT-OPTIONS    S_STDNO    FOR ZTTEST_T-ZZSTDNO
                  MATCHCODE OBJECT ZZMCSBID.
```

```
*-------------------------------------------------------------------*
* INITIALIZATION
*-------------------------------------------------------------------*
INITIALIZATION.

*  数据初始化
  PERFORM DATA_INIT
              USING
                P_CLASS      "班级
                P_YEAR.        "学年
*-------------------------------------------------------------------*
* AT SELECTION SCREEN
*-------------------------------------------------------------------*
AT SELECTION-SCREEN.

*  选择画面项目检查-班级
  PERFORM CLASS_INPUT_CHECK USING P_CLASS TEXT-003 CHANGING TNAME.

*  选择画面项目检查-学年
  PERFORM YEAR_INPUT_CHECK USING P_YEAR TEXT-004.

*-------------------------------------------------------------------*
* AT SELECTION SCREEN ON VALUE-REQUEST
*-------------------------------------------------------------------*
AT SELECTION-SCREEN ON VALUE-REQUEST FOR   P_VARI .

*  选择画面项目选择-布局
  PERFORM VARIANT_INPUT_SELECT.

*-------------------------------------------------------------------*
*START-OF-SELECTION
*-------------------------------------------------------------------*
START-OF-SELECTION.

*  报表数据初始化
  PERFORM REPORT_DATA_INIT.
*  报表数据选择
  PERFORM REPORT_DATA_SELECT.
*  报表数据显示
  PERFORM REPORT_DATA_LIST.

*-------------------------------------------------------------------*
* END-OF-SELECTION
```

```
*---------------------------------------------------------------------*
*END-OF-SELECTION.

* 报表程序终了处理
*   PERFORM REPORT_EDIT_END.
*---------------------------------------------------------------------*
* TOP-OF-PAGE
*---------------------------------------------------------------------*
*TOP-OF-PAGE.

* 报表页眉编辑
*   PERFORM REPORT_HEADER_EDIT.
*---------------------------------------------------------------------*
* END-OF-PAGE
*---------------------------------------------------------------------*
*END-OF-PAGE.

* 报表页脚编辑
*   PERFORM REPORT_FOOTER_EDIT.
*---------------------------------------------------------------------*
* AT LINE-SELECTION
*---------------------------------------------------------------------*
*AT LINE-SELECTION.

* 报表选择处理
*   PERFORM REPORT_LIST_SELECT.

*---------------------------------------------------------------------*
* AT USER-COMMAND
*---------------------------------------------------------------------*
*AT USER-COMMAND.

* 用户交互处理
*   PERFORM REPORT_USER_COMMAND.

*---------------------------------------------------------------------*
* AT PFNN
*---------------------------------------------------------------------*
* AT PFNN.

* 功能键响应处理
*       PERFORM REPORT_PFNN_COMMAND.
*&--------------------------------------------------------------------*
```

```
*&      Form  VARIANT_INPUT_SELECT
*&---------------------------------------------------------------------*
*       选择画面项目选择
*----------------------------------------------------------------------*
FORM VARIANT_INPUT_SELECT .
  AW-VARIANT-REPORT = SY-REPID.        "程序名
  CALL FUNCTION 'REUSE_ALV_VARIANT_F4'
    EXPORTING
      IS_VARIANT              = AW-VARIANT
      I_SAVE                  = AC-A
*      it_default_fieldcat =
    IMPORTING
      E_EXIT                  = AW-EXIT
      ES_VARIANT              = AW-VARIANT
    EXCEPTIONS
      NOT_FOUND = 2.
  IF ( SY-SUBRC = 2 ).
    MESSAGE ID SY-MSGID TYPE CNS-MTY_S  NUMBER SY-MSGNO
             WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ELSE.
    IF ( AW-EXIT = SPACE ).
      P_VARI = AW-VARIANT-VARIANT.
    ENDIF.
  ENDIF.

ENDFORM.                    " VARIANT_INPUT_SELECT
*&---------------------------------------------------------------------*
*&      Form  REPORT_DATA_INIT
*&---------------------------------------------------------------------*
*       报表数据初始化
*----------------------------------------------------------------------*
FORM REPORT_DATA_INIT .

* 内部表初始化
  CLEAR：     STR_TEST.                        "成绩表头
  REFRESH：   TBL_TEST.                        "成绩表
  MOVE：      TEXT-001 TO AW-TITLE ,           "标题
           TEXT-002 TO AW-GRID_TITLE .         "行标题

ENDFORM.                    " REPORT_DATA_INIT
*&---------------------------------------------------------------------*
*&      Form  REPORT_DATA_SELECT
```

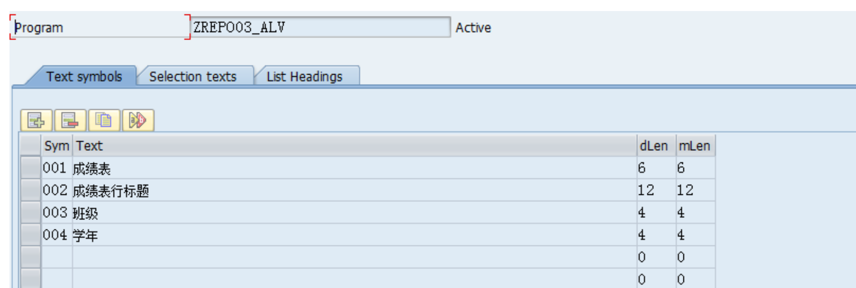
```
*&---------------------------------------------------------------------*
*           报表数据选择
*---------------------------------------------------------------------*
FORM REPORT_DATA_SELECT .
  CREATE OBJECT WO_ALV_OPERATION.

* 设定选择条件
  WO_ALV_OPERATION->CLASS = P_CLASS.
  WO_ALV_OPERATION->YEAR = P_YEAR.
  WO_ALV_OPERATION->STDNO_L = S_STDNO-LOW.
  WO_ALV_OPERATION->STDNO_H = S_STDNO-HIGH.

* 报表数据选择
  TBL_TEST    = WO_ALV_OPERATION->GET_TEST_DATA( ).

ENDFORM.                              " REPORT_DATA_SELECT
*&---------------------------------------------------------------------*
*&           Form    REPORT_DATA_LIST
*&---------------------------------------------------------------------*
*           报表数据显示
*---------------------------------------------------------------------*
FORM REPORT_DATA_LIST .
* 报表数据显示
  WO_ALV_OPERATION->SET_ALV_FULL( TBL_TEST    ).
ENDFORM.                              " REPORT_DATA_LIST
```

文本元素编辑如图 2−21 所示。



Program ZREP003_ALV Active

Text symbols | Selection texts | List Headings

Sym	Text	dLen	mLen
001	成绩表	6	6
002	成绩表行标题	12	12
003	班级	4	4
004	学年	4	4
		0	0
		0	0

图 2−21

选择文本元素编辑如图 2−22 所示。

GUI 状态栏编辑如图 2−23 所示（由标准程序 SAPLKKBL 中的 STANDARD 拷贝而成）。

2.3.2 程序测试

运行程序结果如图 2−24 所示。

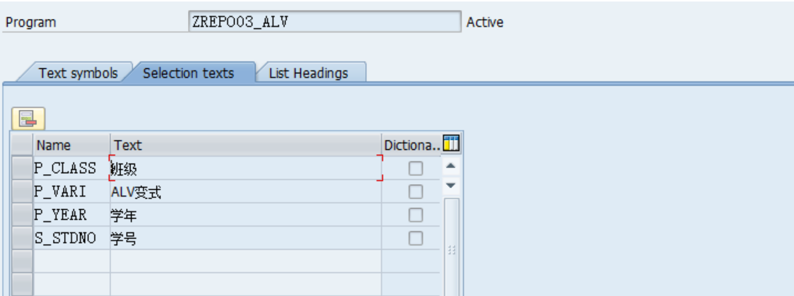


图 2-22



图 2-23

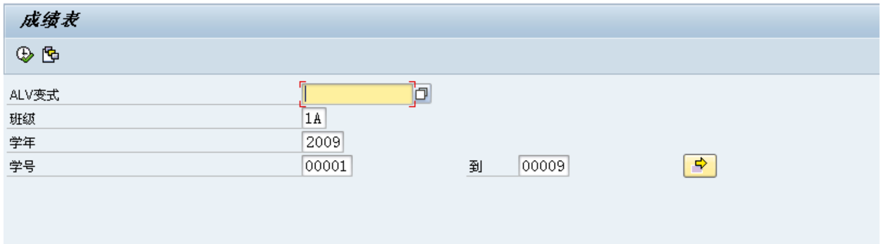


图 2-24

填写相关参数，单击按钮，运行结果如图 2-25 所示。

成绩表

检查表格...

成绩表
作成日 Today:2012/06/15 Time:09:40:36
选择基准 班级:1-A 学年:2009
作成者: YIDATEC

班级	学年	学号	学年	语文成	数学成	自然成	思品成
1A	2009	00001	郑喜定	60	68	70	81
		00002	魏喜奎	62	65	74	83
		00003	金喜善	62	65	71	89
		00004	洪喜官	61	63	75	88
		00005	朴喜珍	64	66	75	88
		00006	王喜凤	64	66	75	88
		00007	刘喜仁	64	66	76	88
		00008	白喜医	62	66	72	88
		00009	赵喜庆	62	63	72	82

图 2-25

ALV 功能点测试结果如表 2-1 所示。

表 2-1

功　能	结　　果
	信息 按钮事件功能代码为 ANL2
双击	信息 双击事件在 行 3 列 ZZTENKOK 上.
热点	信息 金喜善家庭信息是父亲：金魏亭母亲：谢爽英英
	信息 事件 &OUP 触发之前.
	信息 事件 &OUP 触发之后.

2.4　ALV树

2.4.1　程序设计

程序源码设计如下：

```
*---------------------------------------------------------------------*
*
* Program ID                ： ZREPO03_ALV_SIMPLE_TREE
* Program Description       ： 成绩查询ALV 显示
* Create   User             ： BCUSER
* Create   Date             ： 2011/02/20
```

```
*
*-------------------------------------------------------------------*
* Function Overview
*  程序实现以班级、年度、学生为参照进行查询
*    技术点：GUI ALV Tree 形式显示，视图应用
*
*-------------------------------------------------------------------*
* Change List              :
*       change NO.   Change Date     Change User          Change Detail
*       0000001       2011/02/20      BCUSER               程序创建
*-------------------------------------------------------------------*

REPORT ZREPO03_ALV_SIMPLE_TREE
   NO STANDARD PAGE HEADING
   LINE-SIZE 120
   LINE-COUNT 90(3)
   MESSAGE-ID ZTESTMSG.

*-------------------------------------------------------------------*
* INCLUDE
*-------------------------------------------------------------------*
INCLUDE：ZTEST0N,        "常量定义
                ZTEST0F.        "Form 定义

*-------------------------------------------------------------------*
* TYPE-POOLS
*-------------------------------------------------------------------*
TYPE-POOLS SLIS.

*-------------------------------------------------------------------*
* TYPES
*-------------------------------------------------------------------*
*    成绩表
TYPES：  BEGIN OF TYP_TEST,
             ZZCLASS           LIKE ZTTEST_T-ZZCLASS,               "班级
             ZZYEAR            LIKE    ZTTEST_T-ZZYEAR,             "学年
             ZZSTDNO           LIKE ZTTEST_T-ZZSTDNO,               "学号
             ZZNAME            LIKE ZMNUMBER_T-ZZNAME,              "姓名
             ZZTENKOK          LIKE ZTTEST_T-ZZTENKOK,              "语文
             ZZTENSAN          LIKE ZTTEST_T-ZZTENSAN,              "数学
             ZZTENRIK          LIKE ZTTEST_T-ZZTENRIK,              "自然
             ZZTENSHA          LIKE ZTTEST_T-ZZTENSHA,              "思想品德
          END     OF TYP_TEST.

"TYPES    TYP_ITAB_TEST TYPE STANDARD TABLE OF TYP_TEST .
TYPES    TYP_ITAB_TEST TYPE    TYP_TEST OCCURS 0.
```

```
*--------------------------------------------------------------------*
* TABLES：
*--------------------------------------------------------------------*
TABLES：    ZTTEST_T,                                     "考试成绩表
            ZMNUMBER_T.                                   "学生信息表

*--------------------------------------------------------------------*
* STATICS
*--------------------------------------------------------------------*
*STATICS：   XX TYPE X VALUE XX.

*--------------------------------------------------------------------*
* CONSTANTS
*--------------------------------------------------------------------*
CONSTANTS ：
  BEGIN OF AC,
    PROGRAM_NAME     TYPE SY-REPID            VALUE 'ZREPO03_ALV',     "程序
*   控制
    STRUCTURE_NAME TYPE DD02L-TABNAME      VALUE 'ZZTEST_S',         "结构体
    ZZCLASS          TYPE DD03L-FIELDNAME VALUE 'ZZCLASS',           "班级
    ZZYEAR           TYPE DD03L-FIELDNAME VALUE 'ZZYEAR',            "学年
    ZZSTDNO          TYPE DD03L-FIELDNAME VALUE 'ZZSTDNO',           "学号
    ZZTENKOK         TYPE DD03L-FIELDNAME VALUE 'ZZTENKOK',          "语文
    ZZTENSAN         TYPE DD03L-FIELDNAME VALUE 'ZZTENSAN',          "数学
    ZZTENRIK         TYPE DD03L-FIELDNAME VALUE 'ZZTENRIK',          "自然
    ZZTENSHA         TYPE DD03L-FIELDNAME VALUE 'ZZTENSHA',          "思品
    F2CODE(4)        TYPE C                VALUE 'POP',               "F2（双击）
    X(1)             TYPE C                VALUE 'X',
    H(1)             TYPE C                VALUE 'H',
    S(1)             TYPE C                VALUE 'S',
    A(1)             TYPE C                VALUE 'A',
  END OF AC.

*--------------------------------------------------------------------*
* DATA
*--------------------------------------------------------------------*
DATA：
  BEGIN OF TNAME,
    NAME1 LIKE ZMTNUMBER_T-ZTNAME,                      "班主任
    NAME2 LIKE ZMTNUMBER_T-ZTNAME,                      "副班主任
  END OF TNAME.

DATA：    TBL_TEST    TYPE TYP_ITAB_TEST,               "成绩表
          STR_TEST    TYPE TYP_TEST.                    "表头
* ALV 信息定义
```

```
DATA:   BEGIN OF AW,
        FIELDCAT     TYPE   LVC_T_FCAT,                  "字段属性
        FLDCAT_R     TYPE   SLIS_T_FIELDCAT_ALV ,        "字段的内容
        LAYOUT       TYPE   SLIS_LAYOUT_ALV,             "布局
        LISTHEADER      TYPE SLIS_T_LISTHEADER,
        SORT_REC     TYPE   LVC_S_SORT,                  "排序记录
        SORTER       TYPE   LVC_T_SORT,                  "排序属性
        VARIANT      TYPE   DISVARIANT,                  "变式
        GRID_SET     TYPE   LVC_S_GLAY,                  "表格设定
        TITLE        TYPE   LVC_TITLE,                   "成绩表标题
        GRID_TITLE TYPE   LVC_TITLE,                     "成绩表行标题
        EXIT(1)      TYPE   C,
        END OF AW.
* ALV TREE 定义
DATA:   WA_CONTAINER TYPE SCRFNAME VALUE 'ALV_SIMPLE',
        ALV_GRID        TYPE REF TO CL_GUI_ALV_TREE_SIMPLE,
        WA_CUSTOM_CONTAINER TYPE REF TO CL_GUI_CUSTOM_CONTAINER.

*----------------------------------------------------------------------*
* PARAMETERS
*----------------------------------------------------------------------*
PARAMETERS:   P_VARI          TYPE DISVARIANT-VARIANT.        "表示布局（ALV 用）
PARAMETERS    P_CLASS         TYPE ZTTEST_T-ZZCLASS           "班级
                          OBLIGATORY .                        "必须

PARAMETERS P_YEAR TYPE ZTTEST_T-ZZYEAR OBLIGATORY.   "学年

*----------------------------------------------------------------------*
* SELECT-OPTIONS
*----------------------------------------------------------------------*
SELECT-OPTIONS   S_STDNO   FOR ZTTEST_T-ZZSTDNO
                 MATCHCODE OBJECT ZZMCSBID.

*----------------------------------------------------------------------*
* INITIALIZATION
*----------------------------------------------------------------------*
INITIALIZATION.

*  数据初始化
  PERFORM DATA_INIT
              USING
                 P_CLASS          "班级
                 P_YEAR.          "学年
*----------------------------------------------------------------------*
* AT SELECTION SCREEN
*----------------------------------------------------------------------*
```

```
AT SELECTION-SCREEN.

*  选择画面项目检查-班级
   PERFORM CLASS_INPUT_CHECK USING P_CLASS TEXT-003 CHANGING TNAME.

*  选择画面项目检查-学年
   PERFORM YEAR_INPUT_CHECK USING P_YEAR TEXT-004.

*----------------------------------------------------------------------*
* AT SELECTION SCREEN ON VALUE-REQUEST
*----------------------------------------------------------------------*
AT SELECTION-SCREEN ON VALUE-REQUEST FOR    P_VARI .

*  选择画面项目选择-布局
   PERFORM VARIANT_INPUT_SELECT.

*----------------------------------------------------------------------*
*START-OF-SELECTION
*----------------------------------------------------------------------*
START-OF-SELECTION.

*  报表数据初始化
   PERFORM REPORT_DATA_INIT.
*  报表数据选择
   PERFORM REPORT_DATA_SELECT.
*  报表数据显示
   PERFORM REPORT_DATA_LIST.
*  调用画面
   CALL SCREEN '9000'.

*----------------------------------------------------------------------*
* END-OF-SELECTION
*----------------------------------------------------------------------*
*END-OF-SELECTION.

*  报表程序终了处理
*    PERFORM REPORT_EDIT_END.
*----------------------------------------------------------------------*
* TOP-OF-PAGE
*----------------------------------------------------------------------*
*TOP-OF-PAGE.

*  报表页眉编辑
*    PERFORM REPORT_HEADER_EDIT.
*----------------------------------------------------------------------*
* END-OF-PAGE
```

```
*-------------------------------------------------------------------*
*END-OF-PAGE.

*  报表页脚编辑
*    PERFORM REPORT_FOOTER_EDIT.
*-------------------------------------------------------------------*
* AT LINE-SELECTION
*-------------------------------------------------------------------*
*AT LINE-SELECTION.

*  报表选择处理
*    PERFORM REPORT_LIST_SELECT.

*-------------------------------------------------------------------*
* AT USER-COMMAND
*-------------------------------------------------------------------*
*AT USER-COMMAND.

*  用户交互处理
*    PERFORM REPORT_USER_COMMAND.

*-------------------------------------------------------------------*
* AT PFNN
*-------------------------------------------------------------------*
* AT PFNN.

*  功能键响应处理
*       PERFORM REPORT_PFNN_COMMAND.
*&-------------------------------------------------------------------*
*&           Module    INIT_9000    OUTPUT
*&-------------------------------------------------------------------*
*            初期处理
*-------------------------------------------------------------------*
MODULE INIT_9000 OUTPUT.
*  创建 ALV TREE 对象
  IF WA_CUSTOM_CONTAINER IS INITIAL.
    CREATE OBJECT WA_CUSTOM_CONTAINER
      EXPORTING CONTAINER_NAME = WA_CONTAINER.
    CREATE OBJECT ALV_GRID
      EXPORTING I_PARENT            = WA_CUSTOM_CONTAINER.
  ENDIF.
*  显示 ALV TREE 对象
  CALL METHOD ALV_GRID->SET_TABLE_FOR_FIRST_DISPLAY
    EXPORTING
      IT_LIST_COMMENTARY = AW-LISTHEADER
      I_STRUCTURE_NAME      = AC-STRUCTURE_NAME      "参照结构体「ZZTEST_S」
```

```
    CHANGING
      IT_SORT                = AW-SORTER              "初期排序设定
      IT_FIELDCATALOG        = AW-FIELDCAT            "字段目录
      IT_OUTTAB              = TBL_TEST.              "成绩内容
*  展开第一层
  CALL METHOD ALV_GRID->EXPAND_TREE
    EXPORTING
      I_LEVEL = 1.
ENDMODULE.                    " INIT_9000  OUTPUT
*&---------------------------------------------------------------------*
*&      Module  EXIT_COMMAND_9000  INPUT
*&---------------------------------------------------------------------*
*       程序终了处理
*----------------------------------------------------------------------*
MODULE EXIT_COMMAND_9000 INPUT.
  LEAVE TO SCREEN 0.
ENDMODULE.                    " EXIT_COMMAND_9000  INPUT
*&---------------------------------------------------------------------*
*&      Form  VARIANT_INPUT_SELECT
*&---------------------------------------------------------------------*
*       选择画面项目选择
*----------------------------------------------------------------------*
FORM VARIANT_INPUT_SELECT .
  AW-VARIANT-REPORT = SY-REPID.                         "程序名
  CALL FUNCTION 'REUSE_ALV_VARIANT_F4'
    EXPORTING
      IS_VARIANT          = AW-VARIANT
      I_SAVE              = AC-A
*     it_default_fieldcat =
    IMPORTING
      E_EXIT              = AW-EXIT
      ES_VARIANT          = AW-VARIANT
    EXCEPTIONS
      NOT_FOUND = 2.
  IF ( SY-SUBRC = 2 ).
    MESSAGE ID SY-MSGID TYPE CNS-MTY_S  NUMBER SY-MSGNO
            WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ELSE.
    IF ( AW-EXIT = SPACE ).
      P_VARI = AW-VARIANT-VARIANT.
    ENDIF.
  ENDIF.

ENDFORM.                    " VARIANT_INPUT_SELECT
*&---------------------------------------------------------------------*
*&      Form  REPORT_DATA_INIT
```

```
*&---------------------------------------------------------------------*
*            报表数据初始化
*---------------------------------------------------------------------*
FORM REPORT_DATA_INIT .

* 内部表初始化
  CLEAR:       STR_TEST.                              "成绩表头
  REFRESH:     TBL_TEST.                              "成绩表
  MOVE:        TEXT-001 TO AW-TITLE ,                 "标题
               TEXT-002 TO AW-GRID_TITLE .            "行标题

ENDFORM.                          " REPORT_DATA_INIT
*&---------------------------------------------------------------------*
*&          Form    REPORT_DATA_SELECT
*&---------------------------------------------------------------------*
*            报表数据选择
*---------------------------------------------------------------------*
FORM REPORT_DATA_SELECT .

* 成绩表数据抽取
  SELECT        ZZCLASS                 "班级
                ZZYEAR                  "学年
                ZZSTDNO                 "学号
                ZZNAME                  "姓名
                ZZTENKOK                "语文
                ZZTENSAN                "数学
                ZZTENRIK                "自然
                ZZTENSHA                "思品
    FROM ZZTEST_V                       "成绩表
    INTO TABLE TBL_TEST                 "成绩内部表
    WHERE ZZCLASS   =   P_CLASS         "班级
    AND     ZZYEAR = P_YEAR             "学年
    AND      ZZSTDNO    IN S_STDNO.     "学号
* 无数据情况
  IF SY-SUBRC <> 0.
    MESSAGE S002.
    STOP.
  ENDIF.

* 排序
  SORT TBL_TEST BY ZZCLASS ASCENDING                 "班级
                   ZZYEAR ASCENDING                  "学年
                   ZZSTDNO ASCENDING.                "学号

ENDFORM.                          " REPORT_DATA_SELECT
*&---------------------------------------------------------------------*
```

```
*&         Form    REPORT_DATA_LIST
*&---------------------------------------------------------------------*
*           报表数据编辑
*---------------------------------------------------------------------*
FORM REPORT_DATA_LIST .
*  页眉编辑
  PERFORM TOP_OF_PAGE.

*字段清单、字段长度、汇总字段方法调用
*  生成 Field Catalog，

  DATA LS_FLDCAT_R LIKE LINE OF AW-FLDCAT_R.
  DATA LS_FIELDCAT LIKE LINE OF AW-FIELDCAT.
*  声明回调事件结构体
*   DATA LS_EVENT LIKE LINE OF AW-EVENT.

*  功能
*  用来获得程序中的内表 或者 DDIC 表的 Field Catalog
  CALL FUNCTION 'REUSE_ALV_FIELDCATALOG_MERGE'
    EXPORTING
      I_PROGRAM_NAME            = SY-REPID            "PGMID「ZREPO03」
      I_INTERNAL_TABNAME        = 'TBL_TEST'          "设定内部表名
      I_STRUCTURE_NAME          = AC-STRUCTURE_NAME "参照结构体「ZZTEST_S」
*     I_CLIENT_NEVER_DISPLAY = AC-X                   "Client 隐藏
*     I_INCLNAME             =                        "内部表中定义的 Include 名
      I_BYPASSING_BUFFER        = AC-X                "既存的缓存值无效
*     I_BUFFER_ACTIVE          =                      "激活缓存
    CHANGING
      CT_FIELDCAT             = AW-FLDCAT_R           "有文本字段的目录
    EXCEPTIONS
      INCONSISTENT_INTERFACE  = 1
      PROGRAM_ERROR            = 2
      OTHERS                   = 3
            .
  IF ( SY-SUBRC <> 0 ).
    MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
            WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
    EXIT.
  ELSE.
*    编辑 Field Catalog
    LOOP AT AW-FLDCAT_R INTO LS_FLDCAT_R.
      MOVE-CORRESPONDING   LS_FLDCAT_R TO LS_FIELDCAT.
*       设定显示、编辑和合计属性
      IF   LS_FIELDCAT-FIELDNAME = AC-ZZTENKOK     OR      "语文
           LS_FIELDCAT-FIELDNAME = AC-ZZTENSAN     OR      "数学
           LS_FIELDCAT-FIELDNAME = AC-ZZTENRIK     OR      "自然
```

```
          LS_FIELDCAT-FIELDNAME = AC-ZZTENSHA     .           "思品.
        LS_FIELDCAT-OUTPUTLEN = 12.
        LS_FIELDCAT-EDIT           = AC-X.
        LS_FIELDCAT-DO_SUM      = AC-X.
      ELSE.
        LS_FIELDCAT-NO_OUT = AC-X.
      ENDIF.
      APPEND LS_FIELDCAT TO AW-FIELDCAT .
    ENDLOOP.
  ENDIF.
* 创建 TREE 分类字段清单以及先后顺序
*-- 排序字段目录编辑集
*   CLEAR FLDCAT_R.   FLDCAT_R-KEY = ''.
*   MODIFY FIELDCAT FROM   FLDCAT_R TRANSPORTING KEY WHERE KEY = 'X'.
**--默认排序字段设定=> SORTER
*
  CLEAR:   AW-SORT_REC, AW-SORTER.
  AW-SORT_REC-SPOS          = '1'.                  "排序基准「第一」
  AW-SORT_REC-FIELDNAME = AC-ZZCLASS.        "参照字段名「ZZCLASS」
*   AW-SORT_REC-TABNAME     = AC-STRUCTURE_NAME."参照结构体「ZZTEST_S」
  AW-SORT_REC-UP             = AC-X .                  "升序
* AW-SORT_REC-DOWN          = AC-X .                  "降序

  AW-SORT_REC-SUBTOT = 'X'.
  APPEND AW-SORT_REC TO AW-SORTER.
*
  AW-SORT_REC-SPOS          = '2'.                          "排序基准「第二」
  AW-SORT_REC-FIELDNAME = AC-ZZYEAR.                "参照字段名「ZZYEAR」
*   AW-SORT_REC-TABNAME     = AC-STRUCTURE_NAME. "参照结构体「ZZTEST_S」
  AW-SORT_REC-UP             = AC-X .                          "升序

  AW-SORT_REC-SUBTOT = 'X'.
* AW-SORT_REC-DOWN          = AC-X .                          "降序
  APPEND AW-SORT_REC TO AW-SORTER.
*
  AW-SORT_REC-SPOS          = '3'.                          "排序基准「第三」
  AW-SORT_REC-FIELDNAME = AC-ZZSTDNO.              "参照字段名「ZZSTDNO」
*   AW-SORT_REC-TABNAME     = AC-STRUCTURE_NAME. "参照结构体「ZZTEST_S」
  AW-SORT_REC-UP             = AC-X .                          "升序
* AW-SORT_REC-DOWN          = AC-X .                          "降序
  APPEND AW-SORT_REC TO AW-SORTER.
ENDFORM.                              " REPORT_DATA_LIST
*&---------------------------------------------------------------------*
*&          Form   TOP_OF_PAGE
*&---------------------------------------------------------------------*
*           页眉编辑
```

```
*-------------------------------------------------------------------*
FORM TOP_OF_PAGE.
  DATA：  WA_LISTHEADER TYPE SLIS_LISTHEADER,
          H_DATUM(10),
          H_UZEIT(10),
          EMPTY(3)       TYPE C VALUE '   '.
*
  CLEAR WA_LISTHEADER.
* 标题
  WA_LISTHEADER-TYP      = AC-H."'H'.
  WA_LISTHEADER-INFO     = AW-TITLE.
  IF ( NOT AW-VARIANT-TEXT IS INITIAL ).
    CONCATENATE AW-TITLE '(' AW-VARIANT-TEXT ')'
            INTO WA_LISTHEADER-INFO.
  ENDIF.
  APPEND WA_LISTHEADER TO AW-LISTHEADER.
*
  CLEAR WA_LISTHEADER.
  APPEND WA_LISTHEADER TO AW-LISTHEADER.
* 中标题
  WRITE SY-DATUM TO H_DATUM DD/MM/YYYY.
  WRITE SY-UZEIT TO H_UZEIT USING EDIT MASK '__：__：__'.
  CLEAR WA_LISTHEADER.
  WA_LISTHEADER-TYP      = AC-S."'S'.
  WA_LISTHEADER-KEY      =   '作成日'.
  CONCATENATE 'Today：' H_DATUM   EMPTY
              'Time：'   H_UZEIT
          INTO WA_LISTHEADER-INFO .
  APPEND WA_LISTHEADER TO AW-LISTHEADER.

  CLEAR WA_LISTHEADER.
  WA_LISTHEADER-TYP      = AC-S."'S'.
  WA_LISTHEADER-KEY      =   '选择基准'.
  CONCATENATE '班级：' P_CLASS+0(1) CNS-PLUS   P_CLASS+1(1)   EMPTY
            '学年：' P_YEAR
          INTO WA_LISTHEADER-INFO .
  APPEND WA_LISTHEADER TO AW-LISTHEADER.

* 小标题
  CLEAR WA_LISTHEADER.
  WA_LISTHEADER-TYP      = AC-A.
  CONCATENATE '作成者：' SY-UNAME   INTO WA_LISTHEADER-INFO .
  APPEND WA_LISTHEADER TO AW-LISTHEADER.

ENDFORM.                    "TOP_OF_PAGE
```

文本元素编辑如图 2-26 所示。

Program ZREP003_ALV_SIMPLE_TREE Active

Text symbols | Selection texts | List Headings

Sym	Text	dLen	mLen
001	成绩表	6	6
002	成绩表行标题	12	12
003	班级	4	4
004	学年	4	4
		0	0

图 2–26

选择文本元素编辑如图 2–27 所示。

Program ZREP003_ALV_SIMPLE_TREE Active

Text symbols | Selection texts | List Headings

Name	Text	Dictiona..
P_CLASS	班级	☐
P_VARI	ALV变式	☐
P_YEAR	学年	☐
S_STDNO	学号	☐

图 2–27

屏幕 9000 元素清单生成如图 2–28 所示。

屏幕号码 9000 激活

属性 | 元素清单 | 逻辑流

General attr. | Texts/ I/O templates | Special attr. | Display attr. | Mod. groups / functions | References

层.	E	名称	屏幕...	行	列	D...	可...	高度	可...	格式	输入	输出	只...	Di...	词...	Property list
+		ALV_SIMPLE	CCtrl	1	1	86	86	18			☐	☐	☐	☐		
		%#AUTOTEXT001	按	19	83	4	4	1			☐	☐	☐	☐		Properties
		OK_CODE	OK	0	0	20	20	1	☐	OK				☐		

图 2–28

一般属性设定如图 2–29 所示。

控制逻辑编辑如下：

```
************************************************************************
* PROCESS BEFORE OUTPUT
************************************************************************
PROCESS BEFORE OUTPUT.
*  初期处理
   MODULE INIT_9000.

************************************************************************
* PROCESS AFTER INPUT
************************************************************************
```

```
PROCESS AFTER INPUT.

* 程序终了处理
  MODULE EXIT_COMMAND_9000 AT EXIT-COMMAND .
```

屏幕号码 9000 激活

属性 | 元素清单 | 逻辑流

Short description	成绩表		
Original language	ZH 中文	开发类	ZABAP_TRAINING
Last changed on/by	11.06.2015	14:18:02	
Last generation	11.06.2015	14:30:20	

Screen type
- ◉通常
- ○子屏幕
- ○方式对话框
- ○选择屏幕

Settings
- ☐暂存数据
- ☐Switch off runtime compress
- ☐Template - non-executable
- ☐Hold Scroll Position
- ☐Without Application Toolbar

Other attributes

Next Screen		9000	
Cursor position			
Screen group			
Lines/Columns	Occupied	19	86
	Mainten.	27	120
Context menu FORM ON CTMENU			

图 2-29

画面格式编辑如图 2-30 所示。



图 2-30

2.4.2 程序测试

运行程序结果如图 2-31 所示。

填写相关参数，单击按钮，运行结果如图 2-32 所示。

成绩表

ALV变式

班级 1A

学年 2009

学号 00001 到 00009

图 2–31

SAP

成绩表

作成日 Today:2012/07/11 Time:15:29:42

选择基准 班级:1-A 学年:2009

作成者:YIDATEC

班级/学年/学生番号	语文...	数学...	自然...	思品...
1A	561	588	660	775
2009	561	588	660	775
00001	60	68	70	81
00002	62	65	74	83
00003	62	65	71	89
00004	61	63	75	88
00005	64	66	75	88
00006	64	66	75	88
00007	64	66	76	88
00008	62	66	72	88
00009	62	63	72	82

退出

图 2–32

第 3 章　接　　口

随着 SAP R/3 成为国内越来越多大企业 ERP 应用的首选，很多企业都面临着同样一个问题，即如何让新的 ERP 平台和一些老的或是新增加的子系统/子平台默契配合。在保证既有功能的前提下，提高系统和系统间信息传递的效率，从而保证和提高企业的生产竞争力。在这样的背景之下，ERP 系统接口的应用、开发和规范，会起到积极的作用。SAP 从诞生之日开始，就提供了众多的 API 应用编程接口和接口工具，以方便 SAP 二次开发者和爱好者进行研究。RFC、ALE IDocs 是 SAP 公司早期为 SAP R/3R4.6C 版本所提供的接口机制，目前应用最为广泛。

3.1　SAP数据交换接口的类型

SAP 数据交换接口的类型有三种：ALE、RFC 和 CPI-C。它们之间的关系如图 3-1 所示。

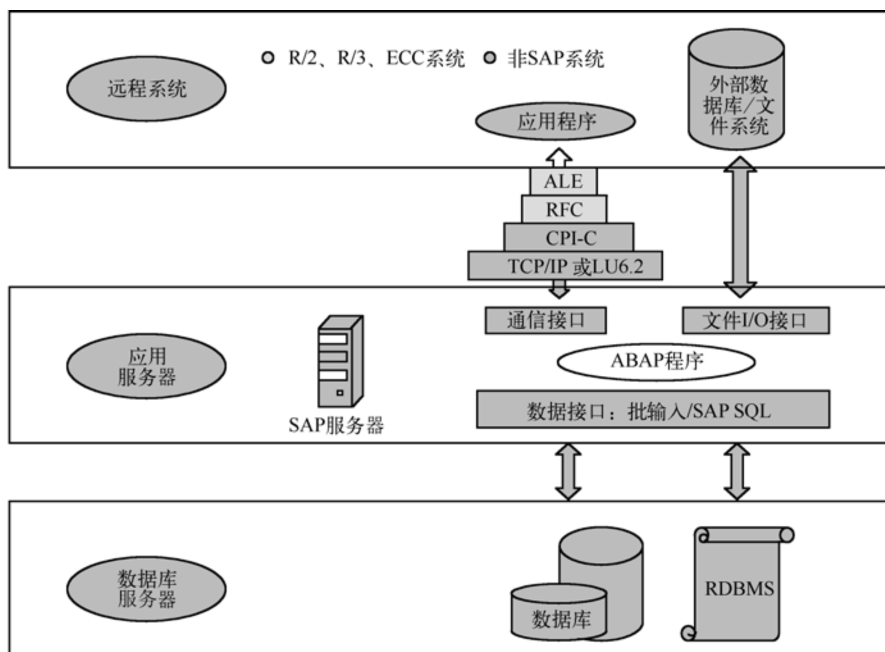


图 3-1

CPI-C 是数据交换的最基本接口，它使用起来比较复杂。

RFC 架构在 CPI-C 接口之上，RFC 的调用都将转换为 CPI-C 的调用完成。

ALE 架构在 RFC 和 IDOC 技术上来完成数据通信交换。

3.1.1　CPI-C简介

CPI-C（Common Programming Interface - Communications）是 SAP 中专门用于程序对程

序的远程调用的协议，其实就是一个定义函数名、参数之类调用需要的信息的数据格式。

CPI-C 是一种同步对话通信模式，参加通信的一方发起一次对话，同时控制信息流动。数据既可以由发送者传递到接收者，也可以反向流动。

参加通信的两个程序需要跟踪对话的状态，如果异常发生导致连接中断，则需要发送方重建并恢复这次通话。通信双方既可以处于主从地位，也可以处于对等地位。也就是说，CPI-C 既支持客户端－服务器环境，也支持对等通信方式。

虽然 CPI-C 在一般情况下是一种同步通信类型，但是在一定环境中，可以通过“临时数据队列”等实现一定程度的异步。TCP/IP 和 SNA 都支持 CPI-C。

由于需要应用程序参与错误的检测与恢复，CPI-C 的编程接口相当复杂，一般情况不用，它只有在以下几种特殊情况下才被采用：

- 客户的系统平台不支持 RFC。
- 旧系统必须使用 CPI-C 作通信接口。
- 系统需要复杂的通信协议。

3.1.2 RFC简介

RFC（Remote Function Call）是 SAP 系统和其他（SAP 或非 SAP）系统间的一个重要而常用的双向接口技术，也被视为 SAP 与外部通信的基本协议。RFC 是 SAP 基于国际标准 CPI-C 之上修改开发的一种 SAP 专用通信协议，用于在不同系统间进行功能调用。简单地说，RFC 过程就是系统调用当前系统外的程序模块，从而实现某个功能的过程，且调用系统和被调用系统中至少有一个必须是 SAP 系统。这种远程功能调用也可在同一系统内部进行（如本地 SAP 系统内的远程调用）；但通常情况下，调用程序和被调用程序处于不同系统。

在多系统、多 Client、多公司的环境下，通常要使用 RFC 技术。RFC 是实现接口的主要方式之一，不但是一种函数，更是一种数据通信协议，类 TCP/IP。

SAP 显然是吃定大集团的管理应用，在大集团通常分散了若干 SAP 应用，可通过 RFC 协议进行连接，主要应用在如下的几个方面：

- MDM 应用：总部 MDM 做整个集团的主数据编码规划，通过 XI+RFC 连接自动分发到各分散服务器。
- BI 应用：数据仓库系统通过 RFC 从分散的 R/3 应用服务器中抽取数据，做报表分析和数据挖掘。
- SLM（Solution Management）应用：SLM 通过 RFC 连接各企业，在 SLM 统一登录，R/3 那边设置好 RFC 用户可自动登录，当然 SLM 还提供了完善的问题处理流程跟踪。

3.1.3 ALE简介

ALE 是 Application Link and Enabling 的缩写，是专门为 SAP 与 SAP 之间所设计的整合中间件。IDocs 是中介文本（Intermediate Document）的缩写，是 SAP 提供的系统整合专用的数据/消息格式，可用于 EDI、ALE 或导入、导出（XML、ASCII）文件等。ALE 从 SAP 3.0 版本开始就作为 SAP 整个应用体系的一部分，为分布式数据交换提供了安全可靠的通信机制。ALE 的设计，原本作为两个 SAP 流程之间的一种消息传递服务（Messaging Service），使 SAP

与 SAP 业务流程之间的企业数据能够有效的交换，为两个独立的 SAP 之间提供了系统的整合服务。不过，随着应用的发展，ALE/IDocs 接口机制也已然成为与其他非 SAP 系统的标准整合方式。

ALE 是工具、程序、数据定义的集合，是为多系统之间的功能实现及数据交互提供的一种机制。

EDI（Electronic Document Interchange，电子数据交换）其实就是采用标准格式的电子数据，用于在通信网络中与业务伙伴交换业务文档。EDI 可以理解为按相同的排列放置数据到一个数据文档中，并按相同的排列解析此文档以得到所需的内容。

EDI 又被叫作无纸化交换，是商业伙伴之间数据交换的一种行业标准，如用户、供应商等。EDI 不同于 ALE，ALE 主要用于同一机构之间的数据交换，而 EDI 主要用于公司之间的数据交换。

IDoc（Intermediate Document，中转文档）一般指具体的 IDocs 数据/消息格式的文本类型，它通过 ALE 方式来进行交换，而 IDoc 提供了 EDI 的支持，也可以把 IDoc 认为是 EDI 的一个实现。

IDoc 在系统间利用消息（Message）传递，不涉及底层函数调用，IDoc 的处理方式是用 EDI 来执行的。

1．数据/消息分配层，主要提供三个关键服务

- 按数据分配模型决定数据接收者。
- 消息的过滤和转换。
- 数据/消息的压缩，以提高传递效率。

2．ALE应用

应用层直接与 SAP 系统接口，生成或从其他系统接收含有路由信息的消息文本 IDocs，包括消息接收者的姓名、要求发送的类型以及对消息进行处理的规则，如图 3−2 所示。

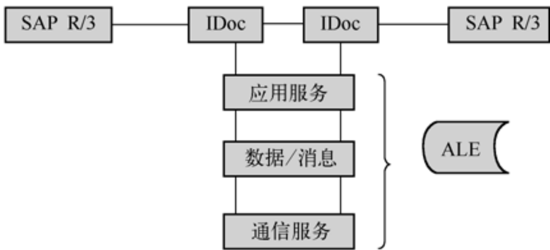


图 3−2

许多大企业把整个 SAP 系统分布在多个实例（Instance）上，每个实例专注于特定的业务。例如，工厂中的物料采购在一个本地的实例上进行处理（本地系统），向供应商付款和总账处理在公司中心实例上处理（中央系统）。分布（Distribute）不是简单的数据分割，而是把数据分布在多个地点，但同时又能保证数据在整体上的一致性和完整性。分布用到分割技术，但是分布不是简单的分割。为了保证不同地点的数据的完整性和一致性，不同系统间利用 ALE 进行沟通。

3.2 SAP R/3 的接口方式

作为目前 ERP 市场上最为领先的应用系统之一，一直以来，SAP R/3 在提供 API 应用编程接口和接口工具方面也同样领先于其他 ERP 厂商。ALE/IDocs 是 SAP 公司为 SAP R/3 R4.6C 版本所提供的接口机制，目前应用最为广泛。在 R4.0 以后的版本中，又添加了技术上先进的 BAPI。本书作为系列介绍之一，对 ALE/IDocs、RFC、BAPI 以及其他可用的整合方式进行介绍。

3.2.1 IDOC接口

IDoc 是 SAP 提供系统集成专用的数据/消息格式。它几乎可以传送任何 SAP 应用数据。IDocs 以文本字符为基础，因而编制方便。IDoc 中的信息从记录类型上分为控制记录、数据记录和状态记录。控制记录主要是文本信息，如 IDoc 类型、发送/接收方信息以及文本标识；数据记录为管理和实际数据部分；状态记录用来追踪文本传递各点的状态，如状态码、系统时间、错误标识等。

适用于中间文件的 IDoc 定义格式，是 SAP 企业系统软件的应用程序之间或 SAP 应用程序与外部程序之间电子数据交换用的标准数据格式。IDoc 是 SAP 的应用程序连接系统的数据转换工具。IDoc 用于数据异步处理：每个 IDoc 生成独立的文本文件，无须连接中央数据库，就可以传送给要求数据的工作平台。SAP 的另一个系统业务应用程序接口则用于数据同步处理。

一个大公司的网络操作环境很可能需要各地分公司的电脑都能与公司的主数据库连接，这些电脑很可能是用不同的硬件或操作系统平台。因为 IDoc 对数据进行了压缩，所以它无须变换格式就能在不同的操作系统上使用。IDoc 类型指定不同种类的数据，比如购买订单或发票，它们可能被划分为更细小的数据种类，即信息类型。更详细的分类意味着一个 IDoc 类型只能储存某一特定交易所需的数据，这样既提高了工作效率又降低了资源损耗。在事务处理过程中，IDoc 随时会生成。例如，在运货交易过程中，可能会产生打印货运清单所需数据的 IDoc。客户在 SAP 系统执行完一项交易后，在数据传送过程中和经过 ALE 通信层时，一个或多个 IDoc 会生成。通信层执行远程功能调用，使用由客户模式规定的端口定义和 RFC 界面定义。IDoc 的接收者可能为 R/3、R/2 或一些外部系统。

1．IDoc数据传输

ALE 的设计结构可以分为三层，即应用层、数据/消息分配层和通信层。通信层是 SAP 整合机制的基础，它利用远程函数调用 RFC（Remote Function Call）调用 SAP 系统的函数模块。

数据/消息分配层，主要提供三个关键服务：

■ 按数据分配模型决定数据接收者。

■ 消息的过滤和转换。

■ 数据/消息的压缩，以提高传递效率。

应用层直接与 SAP 系统接口，生成或从其他系统接收含有路由信息的消息文本 IDocs，包括消息接收者的姓名，要求发送的类型以及对消息进行处理的规则。

ALE 的机制代替了原来的 SAP 所提供的批数据通信 BDC（Batch Data Communication）方式。顾名思义，BDC 为系统之间提供了简单的数据批处理服务，还不能作为一种中间件技术，它没有提供系统之间进行无缝整合所要求的纠错功能、系统管理和其他安全措施。总体说来，应用 SAP 的 ALE 机制进行 SAP 与 SAP 或非 SAP 系统整合有以下几个好处：

1）ALE 技术不受 SAP 版本升级的影响，它提供了版本向后兼容性。ALE 定义于 SAP 应用层，与 SAP 的逻辑层相对独立，整个 ALE 中间件独立于发送和接收系统。

2）ALE 消息设计逻辑保证“一次且只有一次”的消息传递。ALE 采用“存储-发送”技术确保消息可以在系统发生故障或接收方没有准备接收的情况下也可以到达目的地。这样就保证接收方不至于收到重复消息。

3）ALE 也提供了 IDocs 管理功能，主要有文本缩减、文本版本控制以及文本数据过滤。这三种控制机制使得 SAP 开发人员可以根据实际需要对 IDocs 文本在运行中进行动态处理。

4）ALE 还提供了系统管理功能，允许对 ALE 系统进行启动、复位、恢复等系统操作，为开发人员提供了进一步的管理控制。

IDocs 几乎可以传带任何 SAP 应用的数据，是一种“外围”定义格式，与 SAP 的应用数据定义不直接相关。IDocs 已经广泛应用于早期的 SAP-EDI 的数据交换，因而它的设计有点类似于 EDI 的标准，即 EDIFACT 标准。

IDocs 是以字符为基础的，因而是可读的。它有三种记录类型，即：

■ 控制纪录：含文本信息，如 IDoc 类型、发送/接收方信息以及文本标识。

■ 数据纪录：含管理和实际数据部分。

■ 状态纪录：用来追踪文本传递各点的状态，如状态码、系统时间、错误标识等。

2．ALE/IDocs的消息发送接收过程

EDI 的交换有两个流程：

■ 外发（Outbound process 或简称 OP）。

■ 接收（Inbound process 或简称 IP）。

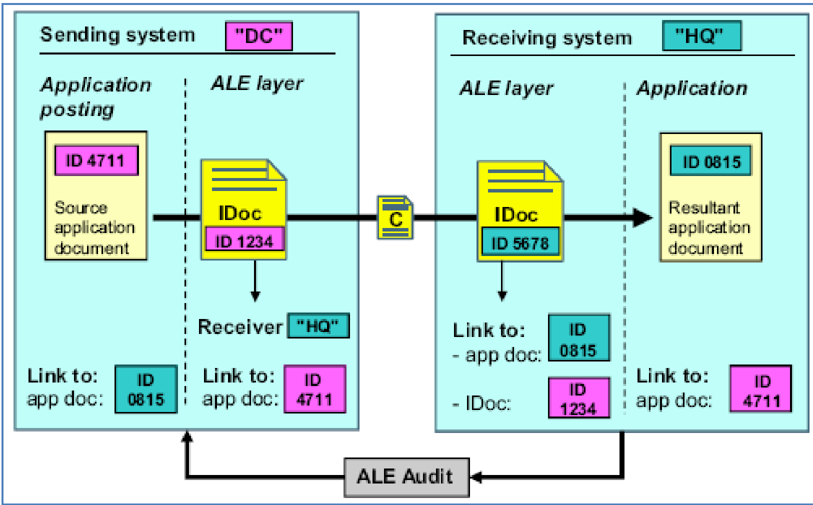


图 3-3

SAP 也是完全遵循着 EDI 的这两个流程，并做了相应的实现。下面对 ALE/IDocs 在系统整合过程中消息的实际传递进行介绍。

■ IDocs 发送过程

一个发送过程由事件触发、文本生成、数据打包以及交由传输媒介传递这四个步骤组成，具体如下：

● 应用层：生成应用数据，应用系统事件触发。

通过应用模块的消息控制间接传送到 IDoc 接口，系统目标（Objects）的状态变化、用户自主活动或其他数据库特定变化等可以启动数据表的触发程序，从而进行数据传递的初始化工作，如数据准备。

● ALE 层：创建出站 IDoc，生成主 IDoc 文本（Master）。

按标准格式生成主 IDoc 文件，包含所有可以传递的数据（不分接收者）。

● IDoc 从 SAP 传送到操作系统。

按照准格式生成主 IDoc 文件被发送到操作系统层。

● 通信层：寻找伙伴和端口生成通信 Idoc。

从主 IDoc 中生成只与特定接收者有关的文本，通信文本是主文本的子数据集（Subset）。

● IDoc 发送

利用异步通信方式将一定版本的 IDoc 传递到接收方。EDI 文件被传送到业务伙伴处（所以业务伙伴可以没有 SAP 系统，因为 EDI 是个标准）。

直接从应用发送到 IDoc 接口，这些处理主要是通过以下函数控制 IDocs 接收过程，如图 3-4 所示：

● 标准 IDoc：MASTER_IDOC_DISTRIBUTE/COMMUNICATION_IDOC_CREATE

● 自定义 IDoc：L_IDOC_SEND/L_IDOC_SEGMENT_CREATE/L_IDOC_HEADER_CREATE



图 3-4

接收过程始于 SAP 系统从外部收到 IDoc 文本。它的优点在于接收方既可以是 SAP 系统，也可以是第三方系统，这也是 SAP 与第三方进行有效整合的基础。接收过程由以下四个步骤组成：

● ALE 层：创建入站 IDoc 并通过伙伴参数文件查找 IDoc 处理类型，存储 Idoc。

将文本存储于数据库，并进行语法校验。

● ALE 层：创建入站 IDoc 并通过伙伴参数文件查找 IDoc 处理类型。

一个专门设计的 IDoc 处理程序读取 IDoc 并产生 SAP 或其他系统所需的系统消息。多个程序可以同时运行。

● IDoc 传送到 SAP 层。

被处理后的 IDoc 被传送到 SAP 系统应用层。

● 应用层：写入文档，生成系统文本。

处理程序进一步生成系统文本供系统使用，并将结果信息存于 IDoc 的状态纪录中。



图 3-5

3．IDoc的特性

每个 IDoc 都被分派了唯一的号码，用于跟踪及以后参考所用。IDoc 包含多个段（Segment），而段内包含有多个字段。IDoc 包含有三种类型的记录：一条控制记录，一条或多条数据记录，一条或多条状态记录。

4．端口（Port）

端口用于外发流程，它判断 EDI 子系统程序名称、IDoc 文件传送到操作系统的目录，IDoc 文件名和 RFC 目的地。RFC 目的地用于定义远程系统通信连接的特性以及需要调用何种功能。

5．Partner Profile

Partner Profile 指定在外发过程中所用的各类组件（业务伙伴号、IDoc 类型、信息类型、端口、处理码等），通信方式（异步或同步）以及当出现错误时通知何人。

6．IDoc的结构

作为一种传输媒介，IDoc 文件的结构包含特定的格式，如图 3-6 所示。

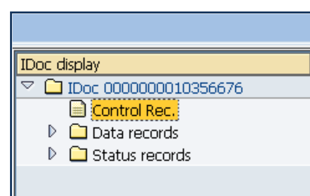


图 3-6

IDoc 结构包括一条控制记录、一条或多条数据记录以及一条或多条状态记录。控制记录（Control Record）也叫 IDoc 头，里面包括 IDoc 的发送者、接收者等信息。数据记录（Data Record）中包括具体的数据信息，比如一个订单的数据信息。状态记录（Status Record）记录 IDoc 从应用到其送达方的当前状态信息，如“已创建”“准备发送”等，是对 IDoc 进行控制的重要信息。

■ 控制记录（Control Record）

每个 IDoc 的 Control Record 可以认为是文件头，它包含发送及接收端、IDoc ID、发送日期等信息，当 IDoc 被触发时，SAP 会首先检查这些信息是否正确，如图 3-7 所示。

■ 数据记录（Data Record）

多个数据段（Segment）组成的一个层次结构，装载传输的数据，且每个数据段不能超过 1000 个字符。当发送或接收这些 IDoc 时，数据将会被解析到这些特定层级关系的数据段中，如图 3-8 所示。

■ 状态记录（Status Record）

用来记录 IDoc 从发送方到接收方的历史状态信息。通常，如果是自行开发的 IDoc，可以在处理 IDoc 逻辑的函数中加入自己定义的状态，以此满足不同的需求，如图 3-9 所示。

注：状态信息由状态维护事务（WE47）维护。



图 3–7

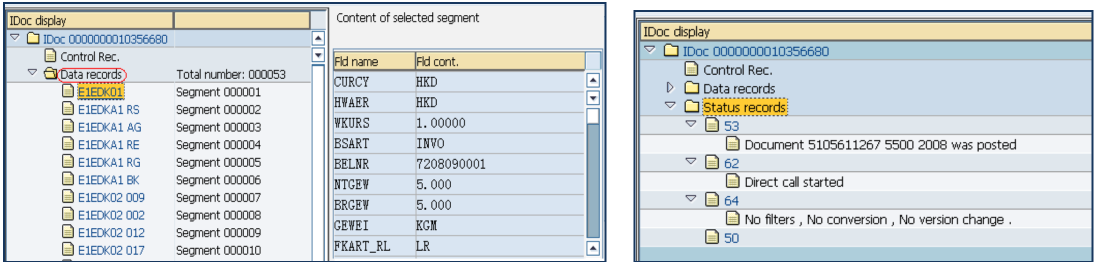


图 3–8　　　　　　　　　　　　　　图 3–9

7．IDoc类型

IDoc 类型是 IDoc 文件的结构定义和说明。在 SAP 中，不同的业务对象有不同的 IDoc 类型，这些类型可以根据不同的传输格式或数据要求多个并存，如图 3–10 所示。

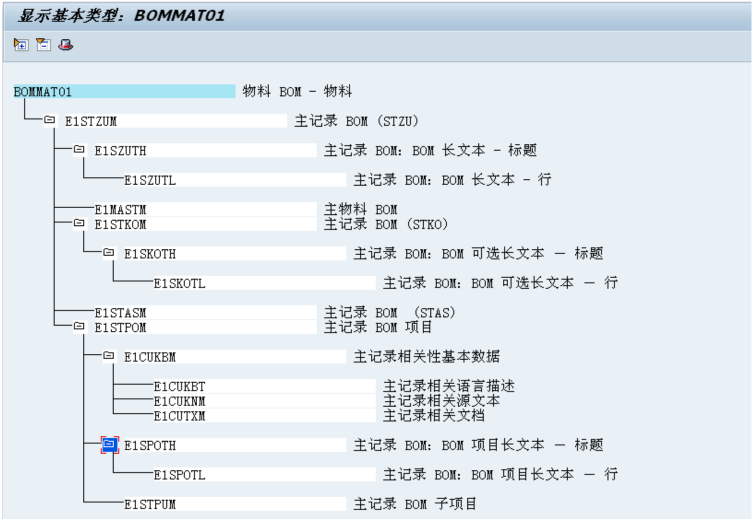


图 3–10

注：IDoc 类型由事务（WE30）维护。

图 3-10 中的示例是自定义委外加工 BOM 的 IDoc 类型定义：
E1STZUM：委外 BOM 的 Header 资料。
E1STPOM：委外 BOM 的 Item 资料。
E1CUKBM 等：每个 Item 的装配组件资料。

8．消息类型（Message Type）

消息类型是指 IDoc 类型的视图，一个消息类型可以包含多个 IDoc 类型。

注：消息类型由事务（WE81 和 WE82）维护。
事务 WE81 用于定义消息类型。
事务 WE82 用于分配 IDoc 类型给消息类型和不同 IDoc 版本（Release）。

9．数据段（Segment）

数据段是 IDoc 的结构组件，这些数据段包含特定的数据类型，每个字段可以参照 SAP 数据字典，也可以自定义字段，保存后系统自动创建版本号，如图 3-11 所示。



图 3-11

维护数据段事务 WE31。

注：IDoc 是基于 SAP 中类似 XML 格式的文档进行数据交换的方式。RFC 则用于在线的方式调用，一般做同步的处理，IDoc 基于文档，可以实现异步的处理。

IDoc 可以理解为是 SAP 为了同外部系统或内部不同 Client 通信所采用的一种数据结构，不同的 IDoc 类型定义了不同的格式，如关于物料主数据的 IDoc，BOM、采购订单、销售订单等相关的 IDoc；ALE 主要用于内部数据交换，如不同 Client；EDI 用于同外部系统的交换数据。它们的本质都是基于 IDoc。

3.2.2 RFC接口

RFC（Romote Function Call：远程函数调用）是 SAP 系统和其他（SAP 或非 SAP）系统间的一个重要而常用的双向接口技术，也被视为 SAP 与外部通信的基本协议。简单地说，RFC 过程就是系统调用当前系统外的程序模块，从而实现某个功能，而且调用系统和被调用系统中至少有一个必须是 SAP ABAP 系统。这种远程函数调用也可在同一系统内部进行（如本地 SAP 系统内的远程调用），但通常情况下，调用程序和被调用程序处于不同系统。

RFC 既是一种接口对象，也是一种特殊的函数模块，可以用 RFC 去调用 SAP 系统里面的 ABAP Function 和外部系统的服务程序。但要实现这种功能必须要有一个 RFC Destination。

SAP 系统 RFC 调用的原理其实很简单，有一些类似于三层构架的 C/S 系统，第三方的客户程序通过接口调用 SAP 内部的标准或自定义函数，获得函数返回的数据进行处理后显示或打印。外部 RFC 调用的模型如图 3-12 所示。

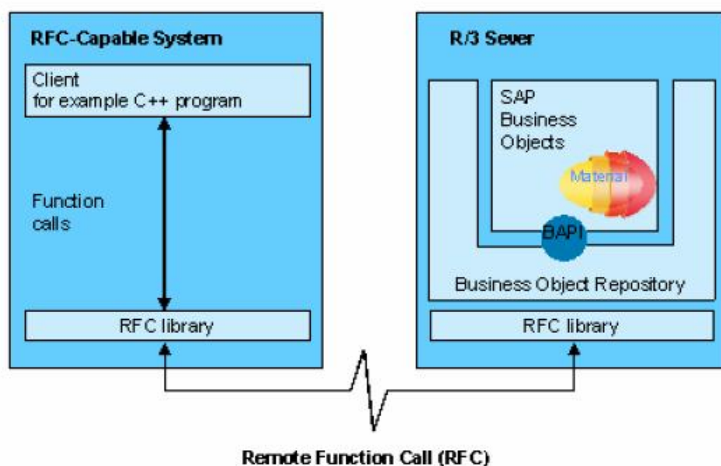


图 3-12

1．RFC的特点

- RFC 是基于 CPI-C 的一种 SAP 接口协议，它大大简化了系统之间的程序交换过程。
- RFC 能自动进行数据传输，当在本地系统相互调用时，则 RFC 相当于一般的函数模型。
- 可以自动登录到与 SAP 相关联的系统，通过设置 RFCDES 先链接外部系统，这时用户名和密码被保存在 RFCDES 表中（事务代码为 SM59）。
- 能管理数据的交换过程，参数传输和错误处理。
- 可以进行远程调试。
- 有 RFC 生成器（RFC Generator）。
- RFC 函数必须为 Remote-Enabled Module，同时 Change Parameter 不可用。
- SAP 公司提供 RFC library（RFC-SDK）为目前所有的外部平台，当外部平台安装完成后，RFC 客户或 RFC 服务器程序就可以通过这个 RFC 函数调用外部程序了。

优点：SAP 的 RFC 调用是其接口技术中最简单和易用的一种方式，该方式开发比较简便，特别适合用于外部报表开发。

缺点：但对于大数据量的查询效率相对较低，一般大数据量开发大多使用 IDoc 和 BAPI 接口开发技术。

2．RFC技术应用

如上所说 RFC（Remote Function Call）远程函数调用，是 SAP 系统之间以及 SAP 与外部系统之间通信的基本接口技术，其过程就是系统调用当前系统外的程序模块，从而实现某个功能，而且调用系统和被调用系统中至少有一个必须是 SAP 系统。

（1）SAP 外部接口

- 程序通信接口
- 文件 I/O 接口
- 数据接口

程序通信接口和文件 I/O 接口是与其他系统进行数据交换的通路，数据接口用来实现 BDC 和 SAP SQL。

（2）RFC 的调用过程

- 在系统间通信过程中，需要区分发送系统和接受系统。
- RFC 调用请求从发送系统中传至接收系统，发送请求的系统在通信过程中称为 RFC 客户端，通信另一方则称为 RFC 服务器。
- RFC 客户端发起远程函数调用以执行 RFC 服务器提供的功能。
- 调用系统和被调用系统均可以是 SAP 系统或非 SAP 系统。

（3）RFC 通信方式（如图 3-13 所示）

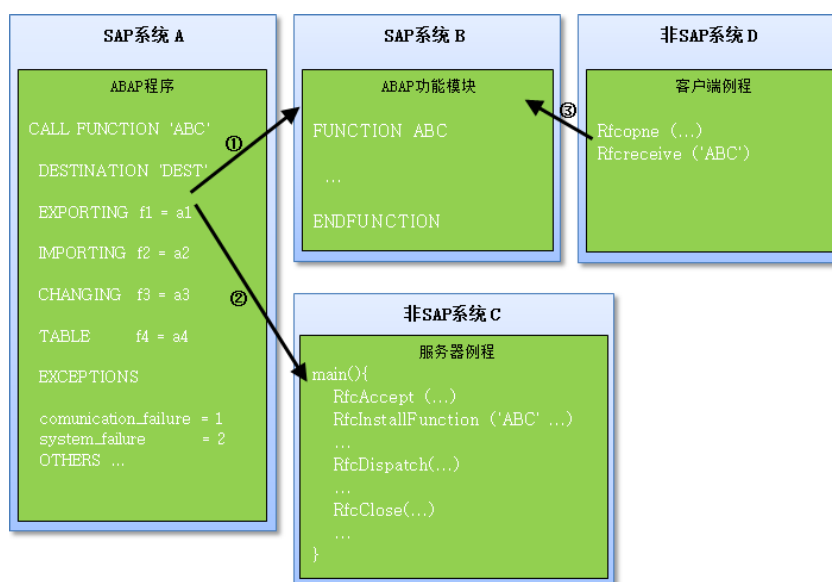


图 3-13

- 两个独立的 SAP 系统之间通信。
- SAP 系统作为调用系统，与外部远程系统（非 SAP ABAP 系统）通信。
- 外部系统作为调用系统，与 SAP 系统通信。在 SAP 系统之间，两个 ABAP 程序通过 RFC 接口进行通信。
- 系统 A 调用 SAP 系统 B 中的功能模块。
- B 系统中被调用的 SAP 功能模块必须支持远程调用，该 SAP 功能模块称为远程函数模块。

在与其他语言的程序进行通信时，SAP 将为各种语言及相关的操作系统提供 RFC 库及 SAP 连接器，以实现 RFC 接口，如：

- 为 C 和 C++提供的 RFC SDK。
- 为 Java 提供的 Java 连接器 JCo。
- 为.NET 提供的.NET 连接器 NCo。

SAP 调用远程功能的能力是通过 RFC 接口系统（RFC Interface System）实现的。

（4）RFC 接口提供以下两种服务

- ABAP 程序的调用接口。

CALL FUNCTION . . . DESTINATION，若远程功能来自 SAP ABAP 系统，则函数必须是实际的函数模块，并支持 RFC 远程调用。

- 非 SAP ABAP 程序的调用接口。

SAP 系统为外部程序提供 RFC 支持接口和 GUI 支持接口，这样在非 SAP 系统中，外部程序就能调用并执行 SAP 远程函数模块，反之亦然。

RFC 调用时，RFC 接口具体功能包括：

- 登录并退出远程系统，并进行权限检查。
- 调用并管理远程系统会话所需的通信进程。
- 自动将实参转换成远程系统所需的格式，RFC 接口支持所有 ABAP 数据类型。
- 处理通信过程中的错误。
- 负责远程调试的实现。

（5）RFC 通信模式

- 同步通信，双方时钟允许误差较小，一次性功能调用。
- 异步通信，双方时钟允许存在一定误差。
- 差异在于调用系统是否需要立即接收返回结果。

SAP 系统中的 RFC 过程扩展有以下几种不同模式：

- 同步 RFC。
- 异步 RFC。
- 事务性 RFC，广义上也属于异步通信模式。
- 队列 RFC，对事务性 RFC 功能的进一步扩展。
- 并行 RFC，实际上是异步 RFC。

（6）支持远程调用的函数模块

- 在创建远程函数模块时，需要在指定属性中的 Processing Type 选项设定为 Remote-Enabled Module。
- 声明参数时，不能在运行时确定参数类型，必须在参数定义时参照数据字典类型进行声明。
- 必须指定为值传（Pass Value），对于表类型参数，系统也隐式进行值传递。
- 格式转换，若传入参数的长度较短，RFC 将在后面补充空格。这意味着对于字符类型数据，长度可以不必完全一致，但是调用方实参不应大于被调用方形参定义的长度。
- 异常处理，可在远程函数中进行异常的抛出，在程序中的异常出现之处通过 raise 语句实现，但是基于 ABAP 类的异常不能从远程函数调用过程中抛回给调用系统，因此需要避免使用。
- RFC 接口预设的系统异常类型：

- SYSTEM_FAILURE，报告远程系统中的所有失败和系统问题。
- COMMUNICATION_FAILURE，若不能建立或通信中连接中断，抛出异常。
- 系统将自动抛出以上 2 种异常，不必在程序代码中进行显示处理，调用 RFC 功能时必须接收，否则出现异常时可能引起系统运行时 SHORT DUMP 中断，所有其他不可知类型的异常指定为 OTHERS 类型。

调用 SAP 标准远程功能：

- 函数模块的调用通过 Call Function 语句实现，远程函数调用，可通过在 Call Function 语句中添加 Destination 子句完成。
- Destination 后面接的是远程目标名称，该目标在 SM59 中设定。
- 可以在远程调用时直接指定当前系统的应用服务器作为 RFC 目标，格式为 <hostname>_<sysid>_<sysnr>。其中<hostname>为主机名，<sysid>为 SAP 系统名，<sysnr>为系统编号。
- 通过 SM51 查看当前 SAP 系统中活动的应用服务器。例如，调用 SAP 标准远程函数模块：SXPG_COMMAND_EXECUTE，通过 RFC 调用时可以运行本地或其他 SAP 系统中的外部命令。

3．RFC的类型

按照 RFC 的调用方式来分，有下面五种类别：

（1）sRFC：同步的 RFC（synchronous RFC）

sRFC 是 RFC 的第一个版本，它要求连接的双方是同步的工作方式，即都是在可用状态才能够实现成功调用。

（2）aRFC：异步的 RFC（Asynchronous RFC）

aRFC 这种 RFC 可以实现异步的 RFC 调用方式，它可以进行多个并发调用，并且不要求被调用系统为可用状态。发出调用系统会一直尝试直到获得被调用系统的应答。它通常用于需要提高系统并行调用多个 RFC 的效率时，相对于强制等待程序的结果，它的效率更高。

（3）tRFC：事务 RFC（transaction RFC）

tRFC 是对 aRFC 进行相关技术改进后的一个 RFC 版本，它与 aRFC 的相同点是实现异步调用，其优点是可以将多个调用进行 LUW 分组处理，并只执行一次运行。现在 aRFC 基本上已经停用。

（4）qRFC：队列 RFC（quene RFC）

qRFC 是 tRFC 的一个增强版本，它保证了所传输数据的处理次序。

（5）pRFC：队列 RFC（parallel RFC）

pRFC 是一种特殊的 RFC，它是 aRFC 的一种扩展类型。因为它改善了系统在执行大量 aRFC 时的性能。SAP 使用它在 MRP 里提高速度，但是它只能执行在同一个系统或同一个 Client 里。

举两个实际的例子：

例子 1，总公司 A 和分公司 C、D、E，某分公司找到新的客户或者客户信息更改时，分公司在本公司系统进行操作，操作完毕后数据通过 RFC 接口更新到总公司系统内，实现信息共享和同步。

例子 2，同样是前面的公司结构，总公司 A 需要对某供应商进行锁定停用或者删除，在 A 公司系统进行操作，操作完成后触发某个远程函数模块（Remote Function Module）将操作同样应用到分公司的系统里，这样总公司锁了的供应商，分公司不能再向它下单了。

4．RFC的原理

（1）同步 RFC

若使用 sRFC，程序就会被同步执行，即主程序会一直等到被 sRFC 调用的程序执行完毕并返回后，才继续往下执行。这样对不同系统之间的数据共享有比较高的安全性，但在程序执行的效率比较低。若 sRFC 是发生在两个不同的 R/3 系统之间，那么可以在主程序那里调试正在运行的远程的被调用程序。还有，在 R/3 系统间的 RFC 里，所有的会话（Dialogs）都会被显示在原调用程序里，如图 3-14 所示。

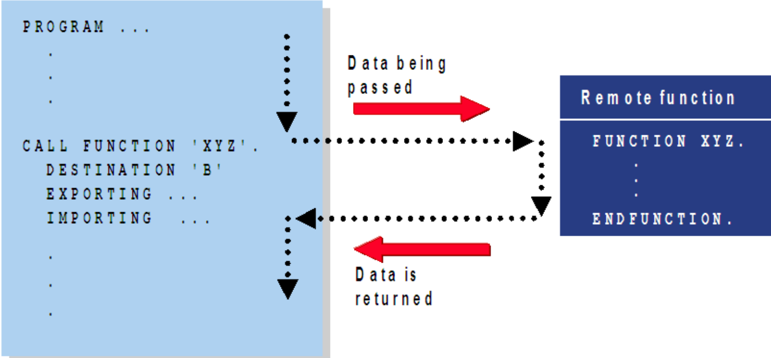


图 3-14

（2）异步 RFC

若使用 aRFC，可以并行执行程序，程序的执行效率比较高，但是数据共享的安全性比较低。因为使用 aRFC 时，被调用的函数可以随时返回数据到原调用程序中，所以原调用程序必须要保持在“Active”状态，直到被调用函数返回为止。并且，当使用 aRFC 时，系统需要保证足够的资源为其他进程服务，如图 3-15 所示。

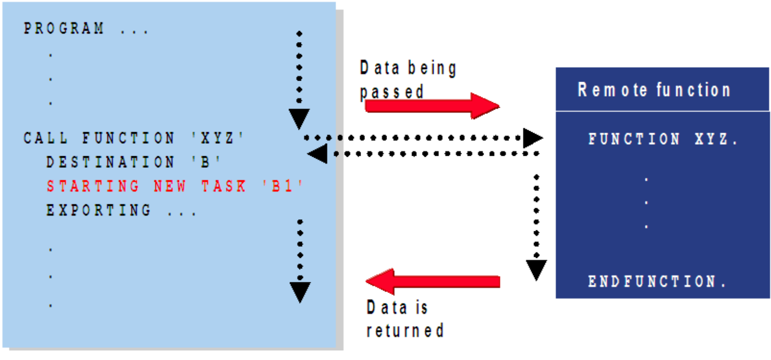


图 3-15

注：同步和异步的区别

■ 同步：提交请求→等待服务器处理→处理完毕返回。这个期间客户端浏览器不能做任何事，如普通 B/S 模式。

■ 异步：请求通过事件触发→服务器处理（这时浏览器仍然可以做其他事情）→处理完毕，如 AJAX 技术。

（3）事务 RFC

tRFC 是 sRFC 和 aRFC 的优点和缺点的折中。tRFC 可以把在同一个 Destination 里的几个函数模块结合在一个单一的逻辑事物单元（Logic Unite of Work）里，然后用事务代码 SM58 去管理 tRFC-LUWs。

在 tRFC 中，利用自动回滚机制，远程地去处理在同一个 LUW 里的几个函数模块。而且，tRFC 能够保证被调用的函数模块被执行，即使在该函数模块所在的系统不能被得到的时候。

实现事务 RFC 必须在 DESTINATION 'NAME'这个语句之前加入 IN BACKGROUND TASK，而且最后 COMMIT WORK 绑定成一个 LUW 发送，如图 3-16 所示。

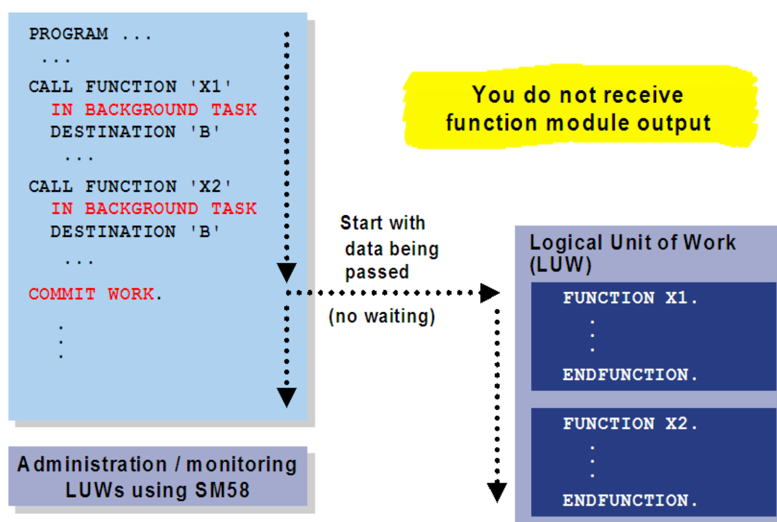


图 3-16

tRFC 与 sRFC 和 aRFC 的不同之处在于远程函数不会返回数据给程序，但可以用事务代码 SM58 去显示或处理（Delect and Execute）有错误或不能被执行的 tRFC-LUWs。

（4）队列 RFC

实际上，qRFC 是把 tRFC-LUWs 连续成一个队列，并能够保证 tRFC-LUWs 按照它们生成时的队列顺序被进行处理。这个队列的属性和 tRFC 输入的数据被保存到队列表 TRFCCOUT 和 tRFC tables ARFCSSTATE/ARFCSDATA 里。可以用事务代码 SMQ1 去显示和维护这些信息。在 Call Function 'NAME' 后加上 Exporting Qname = ' NAME '实现队列 RFC，如图 3-17、3-18 所示。

5．RFC的维护

为了远程调用函数模块，必须要一个进入表 RFCDES 的入口。因为 RFC 远程目的地的信息是在表 FRCDES 里进行维护的，如图 3-19 所示。

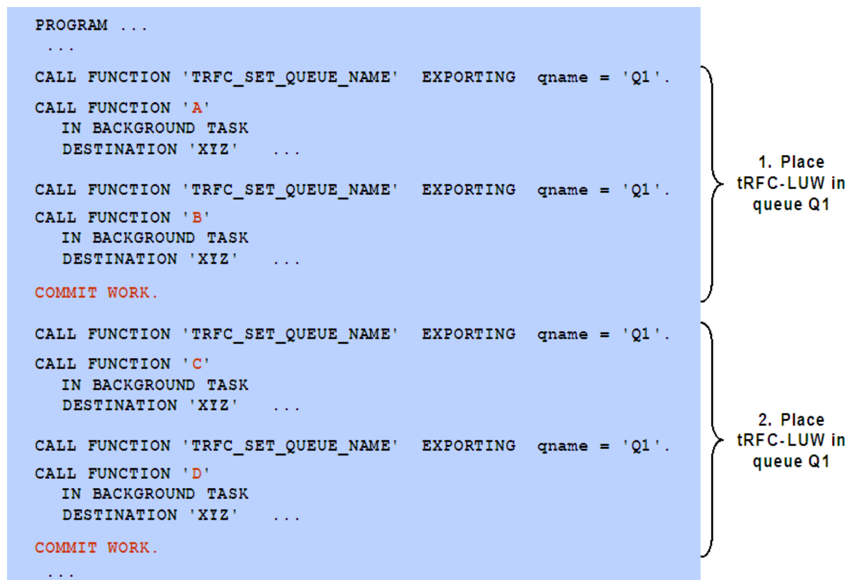


图 3–17

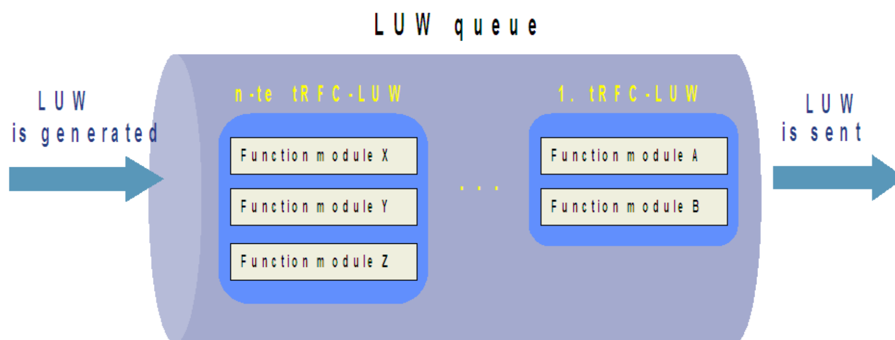


图 3–18

其事务代码是 SM59，如图 3–20 所示。

（1）RFC Destination 主要包括：

■ 类型 2：R/2 连接。

■ 类型 3：ABAP 连接或 R/3 连接，指定主机名和通信服务。

■ 类型 I：内部连接，与当前系统连接到同一 ABAP 系统中，预定义无法修改，与 SM51 中所显示的应用服务器名相同。

■ 类型 L：逻辑目标，通常工作流系统指定过程中配置的 RFC 目标即为该类型的逻辑目标。

■ 类型 X：指定安装了特殊 ABAP 设备驱动程序的系统，必须制定 ABAP 设备驱动程序名。

■ 类型 S：通过 SNA 或 APPC 启动的外部程序连接。

■ 类型 M：通过 CMC 到 ABAP 系统的异步 RFC 连接。

Transp. Table RFCDES Active
Short Description Destination table for Remote Function Call

Attributes | Delivery and Maintenance | Fields | Entry help/check | Currency/Quantity Fields

Srch Help | Predefined Type

Field	Key	Initia..	Data element	Data Type	Length	Decimal..	Short Description
RFCDEST	☑	☑	RFCDEST	CHAR	32	0	Logical Destination (Specified in Function Call)
RFCTYPE	☐	☑	RFCTYPE_D	CHAR	1	0	Type of Entry in RFCDES
.INCLUDE	☐	☐	RFCOPTIONALL	STRU	0	0	RFCOPTION Components of RFCDES Table
RFCOPTIONS	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTIONT	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTIONU	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTIONV	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTION1	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTION2	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTION3	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTION4	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTION5	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTION6	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTION7	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTION8	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTION9	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options
RFCOPTIONA	☐	☐	RFCOPTIONS	CHAR	250	0	Connection options

图 3-19

RFC Connections	Type	Comment
R/2 Connections	2	
ABAP Connections	3	
HTTP Connections to External Server	G	
HTTP Connections to ABAP System	H	
Internal Connections	I	
Logical Connections	L	
TCP/IP connections	T	
Connections via ABAP Driver	X	

图 3-20

■ 类型 T：通过 TCP/IP 并使用 RFC 库或 SAP 连接器的外部程序连接；分为启动（指定主机名、程序路径名）和注册（RFC 服务器程序）两种连接模式。

■ 类型 G：定义外部系统到本地 HTTP 连接。

■ 类型 H：定义 ABAP 系统到本地的 HTTP 连接。

（2）三种通用的有特殊意义的 Destination：

■ Destination 'BACK'：是在两个不同的 R/3 系统之间使用 RFC，如果在被调用的系统的程序里使用了"Destination 'BACK'"的话，那么程序执行到这里就会返回到原调用的系统程序中去。

■ Destination 'SPACE'：是本地调用。

■ Destination 'NONE'：是表示调用通过 Gateway 翻译后，又回到了本地，如图 3-21 所示。（旧的 RFC 使用 SAP CPI-C 协议，必须通过 SAP Gateway 进行翻译，新的 Fast RFC 则不需要使用 CPI-C 协议。）

（3）SAP ABAP 系统间的 RFC 实现（通过 Remote Function Module 实现）

■ 远程调用函数

远程目标可以是文字或变量，其值为与 SAP 系统中一致的远程目标系统。若远程系统是当前系统中的 SAP 应用服务器，也可以直接指定应用服务器名称，如 SM59 中的 I 类型目标。

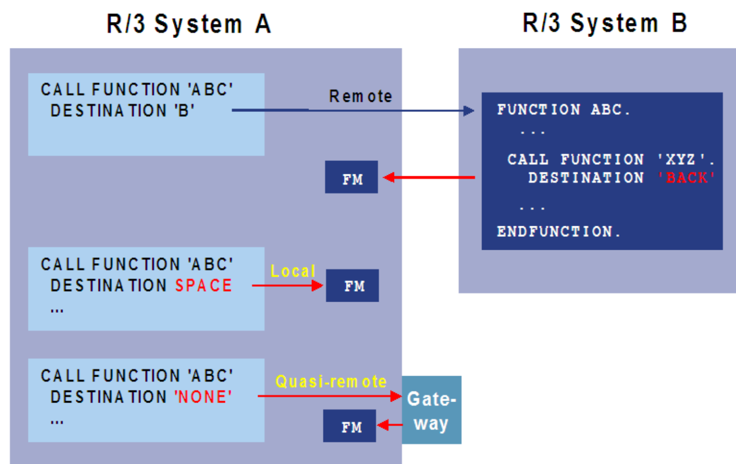


图 3-21

SM59 定义的 RFC 目标是区分大小写的。Destination 附加项中目标变量的值必须与其完全一致。

■ 通过 CALL FUNCTION 语句进行远程功能调用时，可形成不同的调用模式：

● CALL FUNCTION DESTINATION 以同步 RFC 方式实现远程函数调用，若后面无其他附加项，则形成同步远程函数调用，调用程序等待远程调用结果以继续执行。

● CALL FUNCTION STARTING NEW TASK 以异步 RFC 方式实现远程函数调用，调用程序不等待远程调用结果继续执行，结果将在回调子程序（CallBack SubRoutine）中接收。

● CALL FUNCTION IN BACKROUND TASK 以事务性 RFC 方式实现远程函数模块调用，远程功能暂不开始执行，等待 COMMIT WORK 语句出现时，一次性执行一个或多个远程功能。

■ 远程函数参数处理

远程功能调用时，仅允许通过值传递参数，不能进行引用传递，因为在 RFC 过程中，可以传递参数，并返回结果，但不能改变调用程序的上下文。

对表类型参数，在本地普通功能调用中默认为引用传递，不需要创建内表的本地副本，但 RFC 不支持引用传递机制，将进行隐式的值传递调用，必须在 RFC 客户和 RFC 服务器之间交换整个表，只传输实际表格，如果没有指定表参数，则在被调用功能中使用空表。

RFC 使用 delta 管理机制最小化参数和结果传递构成中的网络负载，delta 的意思就是只传递已经变化的数据。

■ RFC 上下文

SAP 程序本地正常情况调用功能模块是将在调用程序的工作过程内部运行（Work Process），但是系统调用远程函数模块时，系统将在独立的工作进程中运行该模块。通过 RFC 接口进行的所有远程功能调用都在目标系统中存在一个独立的 RFC 上下文，即该 RFC 内存滚动区（Roll Area）。

执行 RFC 服务器系统中的远程调用功能后，该功能模块的功能组将被加载至运行时程序

上下文的会话，即功能模块的主程序（或非 ABAP 的 RFC 服务器程序）。

一旦初始化了一个功能模块，程序就将整个功能组载入至主程序的上下文，并保持至调用程序结束。

目标为非 SAP 系统时，该功能通过 RFC 库中的 API 实现（RFCAbort 或 RFCClose），或通过功能模块 RFC_CONNECTION_CLOSE 显示关闭 RFC 连接。

■ 远程函数调用中的 LUM 和隐式数据库提交

SAP 存在特定的数据库提交和回滚机制，该机制就是通过 LUM（逻辑工作单元）实现的。通常情况下的功能模块和主程序位于相同的 LUM 中，因此不会触发数据库提交或回滚的过程（该过程通常通过 COMMIT/ROLLBACK 语句显示触发）。但同步 RFC 和异步 RFC 调用中的每一个功能模块都拥有自己的 LUM，即在每次远程调用时，当前程序都将进行内存区的释放，因此将触发隐式的数据库提交，之前的所有数据库更新将不能回滚，对于事务性 RFC，则有关机制将多个远程调用操作绑定至同一个 LUM，整体提交或回滚。

■ 远程函数调用中的对话处理

对于同步 RFC 和异步 RFC 调用，可以存在对话 dialog 过程，例如 call screen / call transaction 或列表处理等情况，RFC 可以进行后台处理，但如果在后台中心处理 RFC 中的对话，则将导致程序中断，从而触发系统异常 SYSTEM_FAILURE。

在进行 ABAP-ABAP 同步 RFC 和异步 RFC 调用时，可以进行调试以监控远程系统中 RFC 功能执行的情况，可以设置静态断点、单步执行、观察变量等功能，对于同步 RFC，系统在当前会话中进入远程调式界面，对于异步 RFC，系统将打开新会话进行远程调试。

■ RFC 与 Unicode

Unicode 是一种编码体系，建立在 Unicode 编码体系上的系统称为 Unicode 系统，在 Unicode 系统和非 Unicode 系统间进行 RFC 调用时，可能会出现与代码页（code page）转换相关的问题。

■ 同步 RFC 方式的远程函数调用

同步 RFC 要求远程系统在调用时可用，调用程序的处理暂停，并等待远程函数调用的返回结果再继续。

对于表类型参数，只有扁平结构内表可以被传输，其他类型参数可以用于传输纵深类型以及 STRING 类型的数据，将内表通过 tables 参数进行传输将获取较快的传输速度，因为系统内部通过的二进制格式而不是 XML 格式进行传输。

（4）外部程序调用 RFC

■ RFC-SDK（Software Development Kit）必须要安装在外部平台上。

■ 可以用一个外部的 RFC 客户端程序去调用 R/3 系统里的函数模块。

■ 可以从 ABAP 程序里调用 C 程序（这个程序里带有一个已经被声明为远程的 RFC 服务程序）。

■ 在 SAP 系统里，RFC 生成器可以生成一个虚拟的 Function Module 的 C 程序文件集合。这些文件被外部的 RFC 客户端和服务器端使用，如图 3-22 所示。

（5）RFC 的安全性问题

■ 用 RFC 去开始一个 Function Module，那么在目的系统里面，RFC 登录用户要为 Objects_RFC 认证提供一个相应的认证信息。

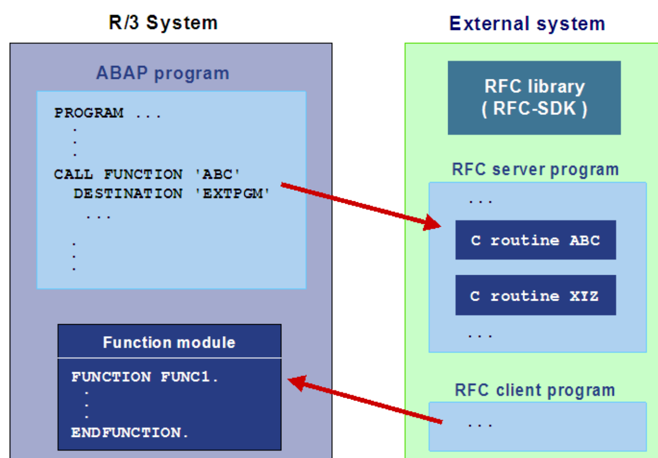


图 3-22

■ 在两个 SAP 系统之间，因为存在一种可信关系，在资源系统里特殊的用户可以使用用户释放功能，而不用 password 就能登录到目标系统的 RFC 里，如图 3-23 所示。

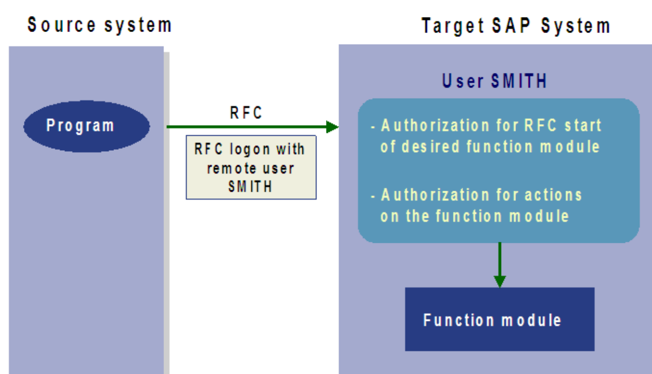


图 3-23

■ 在目标系统里释放一个用户，你需要给认证对象 S_RFCACL 一个认证。

3.2.3 BAPI接口

BAPI 是 Business Application Programming Interface 的缩写，是 SAP 为 3.0 版本以上提供的基于企业目标（Business Object）技术的接口应用界面。

BAPI 是一个标准化的开放接口。它是 SAP 提供给用户的一些标准函数，比如创建销售订单的函数，只要给定相应的参数，就可以实现特定的功能。它也克服了很多 BDC 存在的固有缺陷。

SAP 创建了业务框架，让 SAP 组件之间以及 SAP 和非 SAP 组件之间能够实现技术集成和交换业务数据。业务框架的一个主要组件就是业务应用程序接口（BAPI），它能够表示出组件边界的可视化接口，而属性则把组件集成在一起。

这种集成可以包含本地网络内的组件或者通过互联网相互连接的组件。BAPI 允许在业务

层的集成，而不是技术层。这为连接提供了非常好的可伸缩性和对于未知通信技术的独立性。

SAP R/3 业务框架提供了一个基于 R/3 中功能性应用组件(业务组件)和对象模型的结构。业务框架允许客户和合作伙伴把组件连接到 R/3 系统。面向对象技术和对象模型的使用降低了整个系统的复杂性。

业务框架架构提供了开发 SAP 业务组件的基础，集成和通信架构可以表示如图 3−24 所示。

图 3−24

■ 业务组件

SAP 业务组件提供了自主的业务功能和一致的业务对象。例如，业务对象 Employee 和 Applicant 被分配给业务组件 Human Resource。业务处理或者在业务组件内部实现，或者跨多个组件实现（分布式业务处理）。

■ 业务对象

R/3 系统面向对象的结构基于业务对象。它们封装了业务数据和功能，并且定义了业务组件的功能范围和边界。

■ 业务应用开发接口（BAPI）

BAPI 是业务对象的接口，BAPI 跟业务对象一起从业务层定义和说明了接口标准。

■ 集成服务（Application Link Enable）

ALE 集成服务允许业务处理在不同 R/3 系统或非 R/3 系统上执行的集成。这个使用了 ALE 分布式模型的服务基于业务对象在系统范围上的分布。

■ 通信服务

通讯技术，如分布式组件对象模型（DCOM）和远程函数调用（RFC），它们使用业务框架来访问 BAPI。

1．SAP业务对象

业务对象技术和业务对象编程都基于业务对象的概念。例如，真实世界的对象，一个员工或者一份销售订单，在业务应用系统（如 R/3 系统）中都被模型化为业务对象。建模过程从定义业务对象类型开始。

业务对象类型在 SAP R/3 系统中表示一个业务实体，包含了这个实体的功能（以方法的形式出现）和数据（以属性的形式出现）。业务对象类型的实现细节则对用户隐藏了，业务对象类型通过已经定义好的功能（方法）来访问，这就被称为封装。

业务对象类型用来把 SAP R/3 系统拆分成小的特定单元。由于降低了复杂性，系统的结构也因而改善了。

业务对象类型构成了 SAP R/3 系统的数据和功能的入口。在业务对象类型层，非 SAP 系

统和各种 SAP 业务组件都可以互相通讯。

一个业务对象类型应该可以理解为一个业务概念和它在 SAP R/3 系统中的实现概念。业务对象类型这个术语对应面向对象编程语言中的术语——类。

业务对象类型的某个特定发生的事件，例如实际存储在系统中的一个销售订单，必须要跟业务对象类型本身区分开。这个特定发生的事件被称作一个实例（面向对象编程语言中的常规术语）或者一个业务对象。

2．业务对象类型的结构

图 3−25 描绘了业务对象类型的概念。

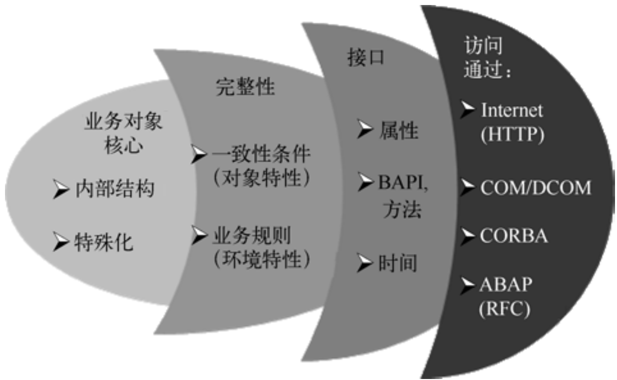


图 3−25

业务对象类型的核心描述了内部对象结构，换句话说就是它的数据和这些数据的结构。约束是确保一个对象一致性的规则（例如，对于不同的最低销售数量可以分配不同的价格条件）。

业务规则要求一个对象必须遵守基本规则，确保对象在它的环境中保持内在的一致性（例如，一个销售组织只能销售给存在业务关系信息的客户）。接口提供了访问业务对象类型的预定义方法，它们可以是公共属性、方法/BAPIA 或事件。

对象可以通过 COM/DCOM 或者 CORBA（通用对象请求代理架构）来访问。在 ABAP 应用中，业务对象可以被远程函数调用（RFC）访问。

3．定义和实现一个对象类型

业务对象类型在业务对象知识库（BOR）中描述，其特性如图 3−26 所示。

■ 基本数据

包含有助于记忆的对象名字，对象类型的技术名称、名称（有意义的说明）、短描述、发布状态、默认方法、默认属性，以及它的实现的程序名。

■ 接口

提供了一个对象预定义的接口（属性、方法、事件），接口类型通常不包含实现。

■ 键字段

只标识唯一一个对象。

■ 属性

表现对象的性质和特征。

■ 方法

封装了 R/3 的功能，可以是同步或异步方法。

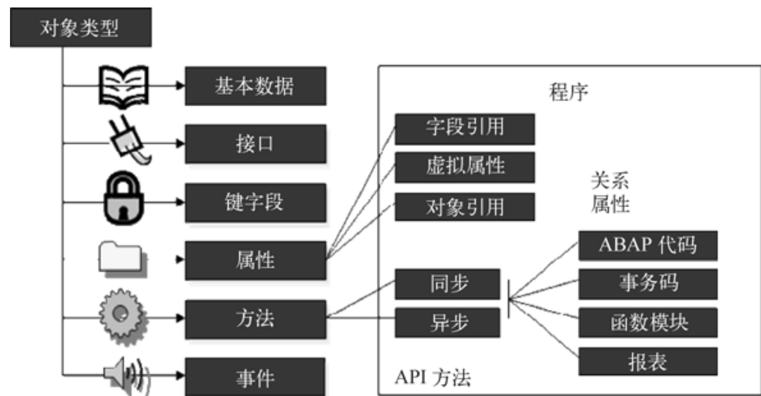


图 3–26

■ 事件

预示业务对象状态改变的事件。

■ 业务对象键字段

每个业务对象类型都有一个或多个键字段。每个实例有一个唯一的键字段的值组合，由键字段中的特定值来标识。这种唯一的值组合机制就能让通过键字段中的值访问一个特定的业务对象类型的实例成为可能。键字段在业务对象设计时是需要重点考虑的。

大多数情况下，对象类型的键字段也是包含了业务对象类型的表头数据表的键字段。因为 BOR 运行时对象是客户端相关的，所以客户端可以不作为一个键字段。

4．业务应用程序接口

业务应用编程接口（BAPI）允许通过形式化、稳定并且无须交互的接口来访问 SAP 的函数。这些接口可以被由客户和其他软件合作伙伴开发的外部应用所使用，也可以被其他 SAP 应用使用。

BAPI 是为业务应用系统精确定义的处理和数据的接口，它作为业务对象知识库（BOR）中的对象的接口来实现。

BAPI 定义成 SAP 业务对象类型的 API 方法。这些对象类型在业务框架中使用，允许组件之间基于对象通信。业务对象和它们的 BAPI 让面向对象可以在公司的中心信息处理中被使用，例如，现存的函数和数据都能被重用，可以达到无障碍的技术协同工作能力，并且可以实现非 SAP 组件。

（1）基本特征

由于 BAPI 作为 R/3 系统的标准接口，它们需要特定的标准、认同和可依赖的特征。在任何开发 BAPI 的时候都必须考虑如下特征：

■ 面向对象：BAPI 作为在业务对象知识库中定义的业务对象的对象方法来实现。

■ 稳定的接口：BAPI 的接口是固定的。

■ 内部和外部使用：BAPI 既可以从 R/3 内部使用，也可以从外部系统调用。

■ 无表示层：BAPI 不提供用户接口功能，显示请求的结果是调用者的职责。

■ 为消息使用参数：成功和错误消息在 RETURN 参数中处理。这可以是个结构或是表，

由 BAPI 来决定。

（2）集成

BAPI 可以用于下列集成类型：

■ 使用 SAP 业务连接器或者互联网应用组件（IAC）把 R/3 连接到互联网上。
■ BAPI 还允许创建真实组件软件，只要它们允许 SAP 组件间的标准通信。最终目标是把 SAP 系统的功能封装到通过通用接口（BAPI）集成的独立业务组件中。
■ 新的 SAP 组件、非 SAP 软件和遗留系统的集成。
■ 实现使用应用连接使能（ALE）异步连接的分布式 R/3 场景。
■ 使用 PC 程序做为 R/3 系统的前台，这可以用 Visual Baisc（Microsoft）或者 Visual Age for Java（IBM）来开发。
■ 扩展超越系统边界的工作流应用。
■ 客户和合作伙伴的自主开发。

（3）集成和可用性目标

当实现 BAPI 时，应该尽量避免以往接口的明信片效应的缺点，即不区分内容和传输。在明信片中，文本写在卡片上，而卡片还充当了信息媒介。如果信息媒介改变了，接收者就会以不同的方式读到文本。这样的结果是，对接口的更改会影响对类型的访问。

在 BAPI 里，清晰地把业务内容与潜在的通信技术区分开来。这个过程可以跟一封在信封中的信来比较：把一封信放到一个新信封里非常容易，同样的，使用一个新的或不同的通信技术，如 COM/DCOM、CORBA 标准或新的互联网标准时也同样容易。信封本身与理解新的内容并不相关。在这个例子中，BAPI 相当于信，也就是说它们与程序语言和通信机制都完全无关。

（4）通信

■ 为了保证它们在任何情形下都能使用，BAPI 必须支持所有通信形式并能被所有类型的参与者使用。
■ 支持同步和异步通信。
  ● BAPI 可以对 SAP 组件进行同步或异步调用。如果调用是同步的，BAPI 就被作为远程函数调用（RFC）。当 BAPI 在发送者系统被调用，就生成一个 IDoc 并分配给接收者系统。当 IDoc 到达接收者，相应 BAPI 的参数就会用 IDoc 数据自动填充，BAPI 再同步调用。
■ 支持机器到机器和人到机器的通信。
■ BAPI 可以用在应用系统集成和连接到各种前端上。
  ● 应用系统的集成可以称为机器到机器的通信，它在交换大量数据和需要很高系统性能时使用。这种情况下的 BAPI 应用将提供很少的细节信息。
  ● 各种前端的集成是人到机器的通信。这时的 BAPI 使用有很多的细节，它们必须设计为允许更灵活的通信以减少错误。
■ 对组件的通信支持被精确连接并通过互联网连接。
  ● 除了在 CORBA 和 COM/DCOM 这样的通信技术下使用 BAPI 外，它还可以通过 SAP 业务连接器在互联网上被访问。业务连接器从 BAPI 调用生成 XML，或者传递一个内含的 XML 文档到 BAPI 调用中。这可以把 BAPI 调用分配成为

XML 文档，允许通过互联网连接的组件之间的通信。

（5）优点

使用 BAPI 具有如下优点：

- BAPI 表现为良好定义、内部一致的单元，总是表示业务事实，并且隐藏的数据库状态一致。
- 业务内容可以被标准化，因为 BAPI 不仅在技术层面上允许 SAP 系统和其他软件的集成，也在业务层面实现了集成。
- BAPI 已经成为业务系统间的通信标准，通过面向对象接口技术（例如 COM/DCOM）的访问已经成为可能。SAP 对象遵守 OAG（Object Application Group）的方针，并且适合于 OMG（Object Management Group）的 CORBA 标准。
- 当 SAP 发布 BAPI 时能够保证其稳定性和兼容性。它的接口定义和参数将长期保持不变，这意味着应用程序即使在 SAP 软件或数据发生更改时也不受影响。如果给 BAPI 添加了向上兼容的增强，现存应用的稳定性也不受影响。
- BAPI 可以支持开放系统，能被任何的开发平台访问。BAPI 是外部系统对 SAP 标准业务进行调用的一种方式，当外部系统有对 SAP 标准业务关联较紧密的功能性开发调用时较适合。

总之、BAPI 是访问 R/3 中业务对象与业务过程的标准编程接口，在 BAPI 定义了 BOR 中业务对象的方法，通过 RFC-Enable 的函数模块实现，用户可以自行开发新的 BAPI 以满足业务需求。BAPI 是 SAP 提供的基于业务对象的函数，它处理的对象是 R/3 的业务相关 Business Object，比如单据类销售订单、组织、公司等，它们是一系列实体。RFC 则是一种系统间通信的方式（Remote Function Call），但当外部系统与 SAP 系统间有大数据量的交付，并且分布广、网络状况较差的情况下不适合。

3.3 ALE/IDoc应用

IDoc 是信息系统之间（通常是 ERP）进行事务性或数据交换使用的标准数据格式，其中比较公认的是符合 EDI 标准的 IDoc，个别系统可以用 EDI 子系统转换为 EDI 标准来使用。IDoc 应用一般是面向消息的，含有事务性数据、主数据和控制数据处理。

常见的应用如 SAP 系统之间通过 IDoc 共享各系统间的主数据，从外部系统接收 IDoc 更新事务数据等（一般要求 SAP 系统间配置信息一致），ALE 配置用事务代码：SALE，如图 3−27 所示。

3.3.1 SAP系统间数据传输配置

本例中实现物料数据共享，由一方系统创建物料，通过 IDoc 发送与接收操作，在另一方系统中实现共享。

1．定义系统间连接

（1）在接收方系统定义

步骤一：定义发送方与接受方逻辑系统。

按照路径“IDoc 接口/启用应用程序链接（ALE）→基本设置→逻辑系统→定义逻辑系

统”或运行事务代码 BD54，添加逻辑系统如图 3–28 所示。



图 3–27

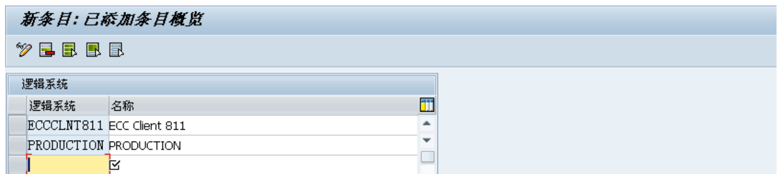


图 3–28

如上图所示，逻辑系统 ECCCLNT811 为接收方用，PRODUCTION 为发送方用。

步骤二：定义客户端与逻辑系统之间的对应关系。

运行事务代码 SCC4，设定客户端信息如图 3–29 所示。

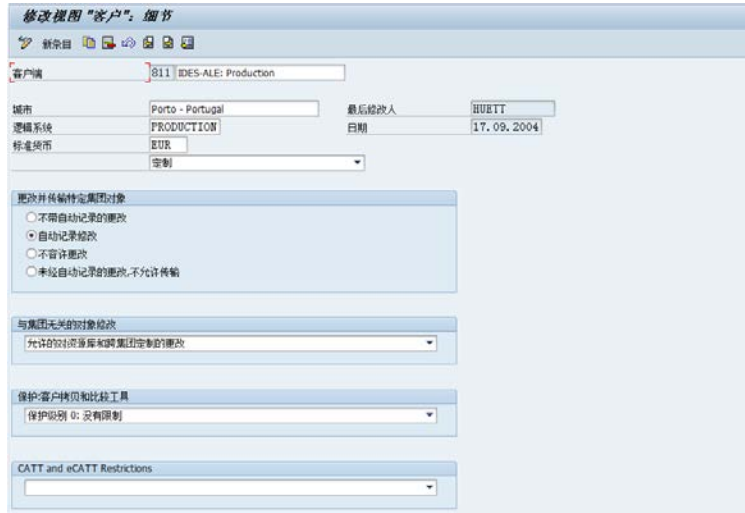


图 3–29

步骤三：定义 RFC 调用目标系统，建立连接。

运行事务代码 SM59，设定 RFC 连接信息如图 3–30 所示。

图 3–30

定义接受方 RFC 连接：填入接收方目标系统名称、IP 地址和实例编号。

输入语言、客户端号、用户名和密码，如图 3–31 所示。

图 3–31

选择“MDMP & Unicode”选项，如图 3-32 所示。

RFC终点ECCCLNT811

远程登录 连接 Unicode Test

RFC 目标系统 ECCCLNT811

连接类型 3 R/3连接 Description

说明

Description 1 ECCCLNT811

Description 2

Description 3

Administration | Technical Settings | 登录 | MDMP & Unicode | Special Options

Communication Type with Target System

Non-Unicode

MDMP Settings

Inactive Active

Unicode

Character Conversion

Default Setting '#' = U+0023

Short Dump After Conversion Error

Ignore Conversion Errors

Display of Conversion Errors

Error Indicator #

U+ 0023

图 3-32

测试连接和权限测试，单击应用工具栏中的 连接 按钮，如图 3-33 所示。

R F C-连接测验

连接测试 ECCCLNT811

连接类型 R/3 连接

Aktion	Ergebnis
注册：	23 msec
Übertragung 0 KB:	0 msec
Übertragung 10 KB:	1 msec
Übertragung 20 KB:	1 msec
Übertragung 30 KB:	2 msec

图 3-33

注：此处测试连接时一般要单击 远程登录 按钮，查看是否能远程登录到对方系统，如图 3-34 所示。

定义发送方 RFC 连接：填入发送方目标系统名称、IP 地址和实例编号，如图 3-35 所示。

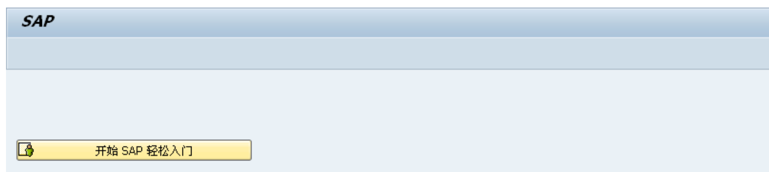


图 3-34

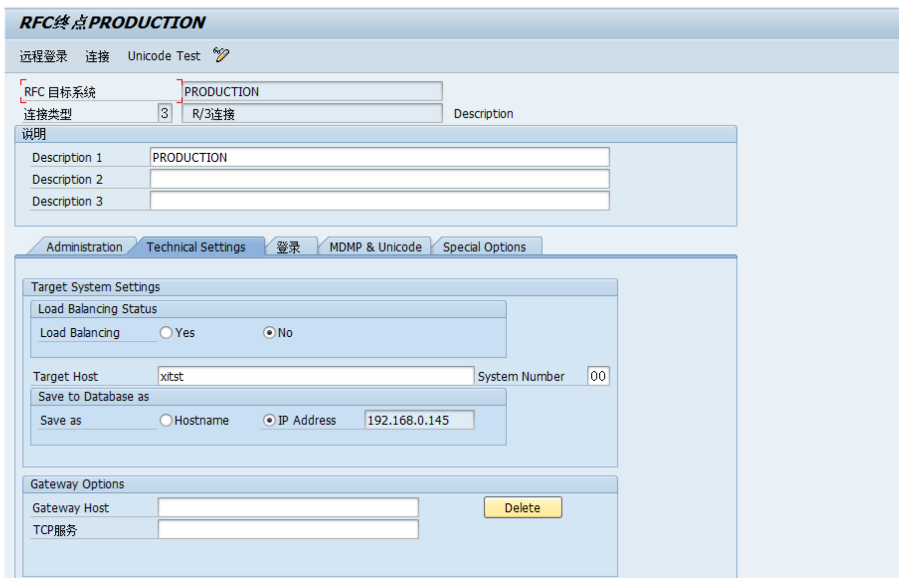


图 3-35

输入语言、客户端号、用户名和密码，如图 3-36 所示。

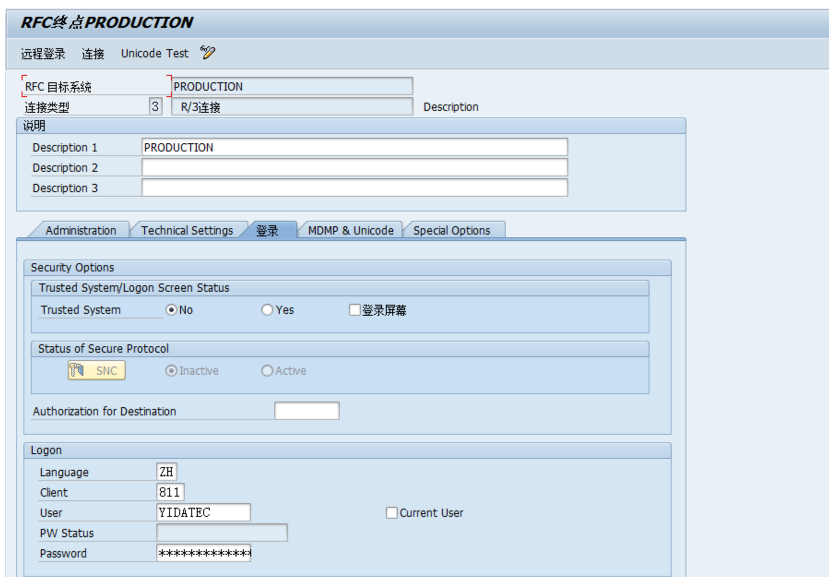


图 3-36

选择 Unicode 选项，如图 3-37 所示。

图 3-37

注：当对方系统为 Unicode 系统时，该选项要被选中。

测试连接和权限测试，单击应用工具栏中的 连接 按钮，如图 3-38 所示。

图 3-38

（2）在发送方系统定义

按照以上步骤，在发送方定义相同的逻辑系统和对应的 RFC 连接。

2．定义系统间IDoc连接

（1）在接收方系统定义

步骤一：创建并分配添加消息类型。

运行事务代码 BD64，创建模型视图，如图 3-39 所示。



图 3-39

单击按钮并改为修改模式，单击 创建模型视图 按钮创建模型视图，输入描述与 IDoc 名称，如图 3-40 所示。

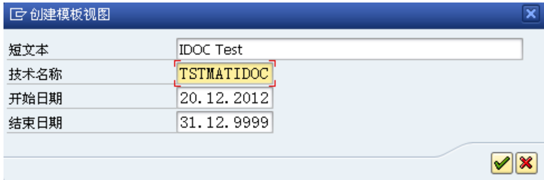


图 3-40

单击上图中的按钮，效果如图 3-41 所示。



图 3-41

选中模型视图，单击[添加消息类型]按钮添加消息模型，输入模型名称、发送方逻辑名、接受方逻辑名和消息类型如图 3-42 所示。

单击上图中的✔按钮，并单击标准工具栏上的保存按钮保存设定信息。

步骤二：生成伙伴参数文件。

按照以下路径生成伙伴参数文件，如图 3-43 所示。

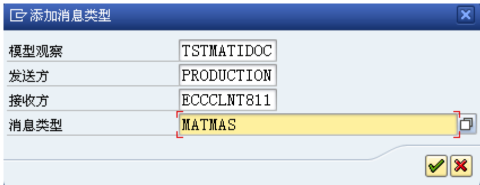


图 3-42

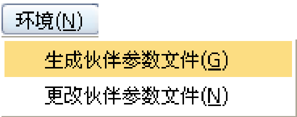


图 3-43

输入合作伙伴系统（发送方逻辑系统名称），如图 3-44 所示，单击运行按钮。

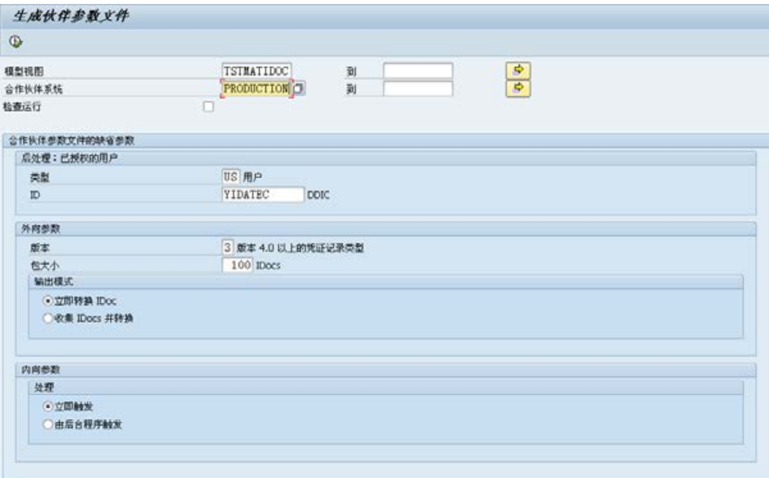


图 3-44

执行结果如图 3-45 所示。

生成伙伴参数文件

用于生成伙伴参数文件的日志	
伙伴	
系统 ECCCLNT811	伙伴 ECCCLNT811 作为伙伴已经被创建
系统 PRODUCTION	系统 PRODUCTION 作为伙伴类型已经存在
端口	
系统 PRODUCTION	带有 RFC 目标 A000000004 的端口 IDES_Central 已经存在
外向参数	
系统 PRODUCTION	消息类型为SYNCH SYNCHRON 的外向参数成功创建
内向参数	
系统 PRODUCTION	消息类型为 MATMAS 的内向参数已成功创建

图 3-45

步骤三：分配观察模型。

单击标准工具栏上的返回按钮，按照以下路径将模型分配给逻辑系统，如图 3-46 所示。

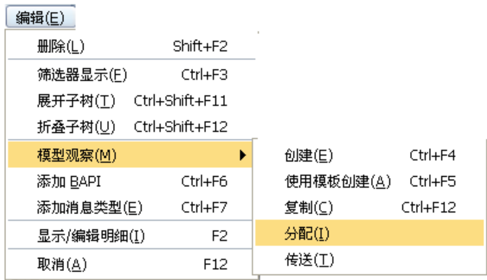


图 3-46

选择发送方逻辑名称，如图 3-47 所示。

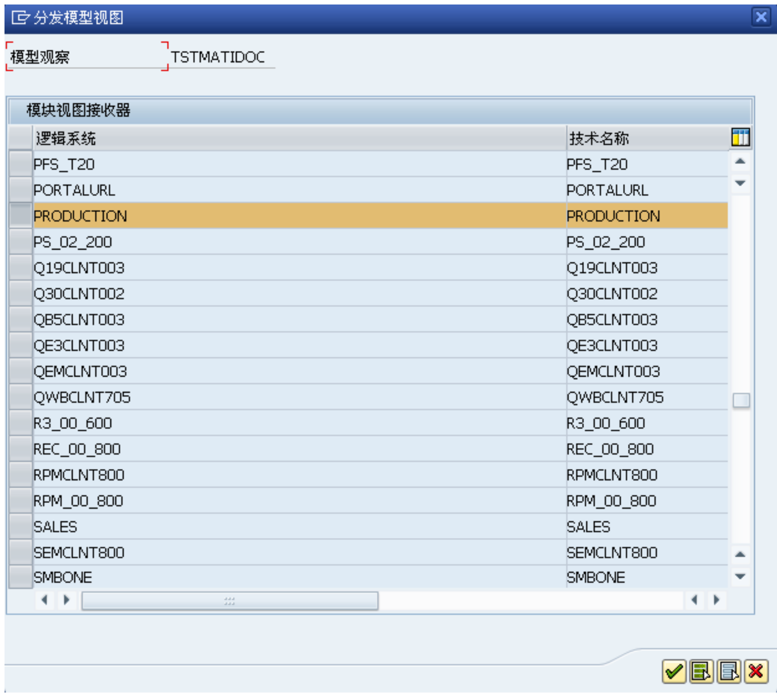


图 3-47

单击上图中的按钮，分配结果如图 3-48 所示。



图 3-48

接收方配置结果如图 3-49 所示。

显示分配模型

筛选器模型显示　创建模型视图　添加 BAPI　添加消息类型

分配模型	描述/技术名	业务对象
模型视图		
AL	AL　　. 不存在短文本	
ALE-AUDIT	ALE-AUDIT . 不存在短文本	
CONSOLIDAT	CONSOLIDAT. 不存在短文本	
CONTRLDATA	CONTRLDATA. 不存在短文本	
EBP_TO_R3	EBP_TO_R3 . 不存在短文本	
LOG-HR	LOG-HR　. 不存在短文本	
MDM30	MDM30　. 不存在短文本	
MDMDEMO811	MDMDEMO811. 不存在短文本	
MDM_001	MDM_001　. 不存在短文本	
MDM_002	MDM_002　. 不存在短文本	
MDM_003	MDM_003　. 不存在短文本	
MDM_004	MDM_004　. 不存在短文本	
MDM_005	MDM_005　. 不存在短文本	
MDM_006	MDM_006　. 不存在短文本	
MDM_007	MDM_007　. 不存在短文本	
MDM_008	MDM_008　. 不存在短文本	
MDM_009	MDM_009　. 不存在短文本	
MDM_010	MDM_010　. 不存在短文本	
SUBSYSTEMS	SUBSYSTEMS. 不存在短文本	
IDOC Test	TSTMATIDOC	
PRODUCTION	PRODUCTION	
ECC Client 811	ECCCLNT811	
MATMAS	物料主数据	
没有设置筛选器		

图 3-49

发送方生成结果如图 3-50 所示。

显示分配模型

筛选器模型显示　创建模型视图　添加 BAPI　添加消息类型

分配模型	描述/技术名	业务对象
模型视图		
AL	AL　　. 不存在短文本	
ALE-AUDIT	ALE-AUDIT . 不存在短文本	
CONSOLIDAT	CONSOLIDAT. 不存在短文本	
CONTRLDATA	CONTRLDATA. 不存在短文本	
EBP_TO_R3	EBP_TO_R3 . 不存在短文本	
LOG-HR	LOG-HR　. 不存在短文本	
MDM30	MDM30　. 不存在短文本	
MDMDEMO811	MDMDEMO811. 不存在短文本	
MDM_001	MDM_001　. 不存在短文本	
MDM_002	MDM_002　. 不存在短文本	
MDM_003	MDM_003　. 不存在短文本	
MDM_004	MDM_004　. 不存在短文本	
MDM_005	MDM_005　. 不存在短文本	
MDM_006	MDM_006　. 不存在短文本	
MDM_007	MDM_007　. 不存在短文本	
MDM_008	MDM_008　. 不存在短文本	
MDM_009	MDM_009　. 不存在短文本	
MDM_010	MDM_010　. 不存在短文本	
SUBSYSTEMS	SUBSYSTEMS. 不存在短文本	
IDOC Test	TSTMATIDOC	
XIP 集团 811	PRODUCTION	
ECCCLNT811	ECCCLNT811	
MATMAS	物料主数据	
没有设置筛选器		

图 3-50

（2）在发送方系统定义

步骤一：生成伙伴参数文件。

运行事务代码 BD64，选中已生成的模型，按照以下路径生成伙伴参数文件，如图 3–51 所示。

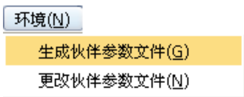


图 3–51

输入合作伙伴系统（接收方逻辑系统名称）如图 3–52 所示，单击运行按钮。

生成伙伴参数文件

模型视图	TSTMATIDOC	到	
合作伙伴系统	ECCCLNT811	到	
检查运行	☐		

合作伙伴参数文件的缺省参数

后处理：已授权的用户

类型	US 用户
ID	YIDATEC DDIC

外向参数

版本	3 版本 4.0 以上的凭证记录类型
包大小	100 IDocs

输出模式

- ◉ 立即转换 IDoc
- ○ 收集 IDocs 并转换

内向参数

处理

- ◉ 立即触发
- ○ 由后台程序触发

图 3–52

显示成功信息如图 3–53 所示。

生成伙伴参数文件

用于生成伙伴参数文件的日志

伙伴	
系统 ECCCLNT811	伙伴 ECCCLNT811 作为伙伴已经被创建
系统 PRODUCTION	系统 PRODUCTION 作为伙伴类型已经存在
端口	
系统 ECCCLNT811	带有 RFC 目标 A000000016 的端口 ECCCLNT811 已经创建
外向参数	
系统 ECCCLNT811	消息类型为MATMAS MATMAS05 的外向参数成功创建 消息类型为SYNCH SYNCHRON 的外向参数成功创建

图 3–53

3．测试系统间IDoc连接

（1）在发送端发送

步骤一：创建物料。

运行事务代码 MM01，创建物料如图 3–54 所示。

图 3–54

步骤二：发送物料。

运行事务代码 BD10，填写发送的物料号和接收方服务器逻辑地址，单击运行按钮，如图 3–55 所示。

图 3–55

运行结果如图 3–56 所示。



图 3-56

单击上图中的按钮，如图 3-57 所示。



图 3-57

单击上图中的按钮。

步骤二：查看发送物料结果。

发送完成后，运行事务 WE02 查看发送消息，如图 3-58 所示。

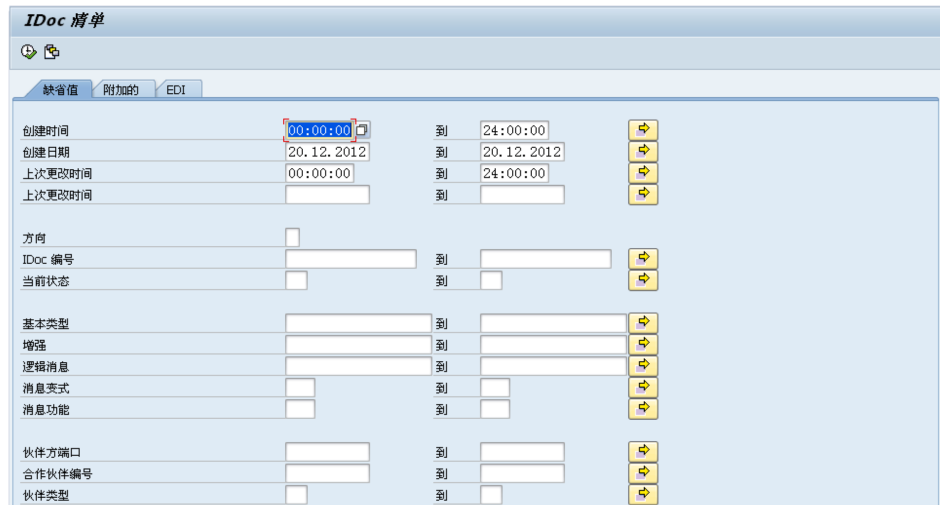


图 3-58

单击运行按钮，如图 3-59 所示。

（2）在接收端接收

步骤一：查看接受消息。

在接受方服务器运行事务 WE02，查看接受消息，如图 3-60 所示。

步骤二：查看接受物料。

在接受方服务器运行事务 MM03，查看接受物料，如图 3-61 所示。



图 3-59



图 3-60



图 3-61

3.3.2 SAP系统间数据扩张传输

本例中实现传送自定义表中的数据共享，一方系统表中的数据，通过 IDoc 发送与接收操作，在另一方系统中实现共享。

1．确认系统数据状况

发送端系统逻辑系统名称为 NSPCLNT000（客户端为 000），接收端系统逻辑系统名称为 PRODUCTION（客户端为 811）。

传送的表结构如图 3-62 所示，此表在 NSPCLNT000 和 PRODUCTION 中均存在。

转帐表 ZTTEST_T 激活
短文本 考试成绩表

Attributes | Delivery and Maintenance | Fields | Entry help/check | Currency/Quantity Fields

Srch Help　Predefined Type

	Key	Initi...	数据元素	数据类型	长度	小数位	短文本
MANDT	☑	☑	MANDT	CLNT	3	0	客户端
ZZCLASS	☑	☑	ZZCLASS_E	CHAR	2	0	班级
ZZYEAR	☑	☑	ZZYEAR_E	NUMC	4	0	学年
ZZSTDNO	☑	☑	ZZNUMBER_E	CHAR	5	0	学生番号
ZZTENKOK	☐	☐	ZZTENKOK_E	DEC	3	0	语文成绩
ZZTENSAN	☐	☐	ZZTENSAN_E	DEC	3	0	数学成绩
ZZTENRIK	☐	☐	ZZTENRIK_E	DEC	3	0	自然成绩
ZZTENSHA	☐	☐	ZZTENSHA_E	DEC	3	0	思品成绩
.INCLUDE	☐	☐	ZTLOG_S	STRU	0	0	表修改日志
ZZIUSER	☐	☐	ZZIUSER_E	CHAR	12	0	创建人
ZZIDATE	☐	☐	ZZIDATE_E	DATS	8	0	创建日期
ZZITIME	☐	☐	ZZITIME_E	TIMS	6	0	创建时间
ZZCUSER	☐	☐	ZZCUSER_E	CHAR	12	0	修改人
ZZCDATE	☐	☐	ZZCDATE_E	DATS	8	0	修改日期
ZZCTIME	☐	☐	ZZCTIME_E	TIMS	6	0	修改时间
ZZDELETE	☐	☐	ZZDELETE_E	CHAR	1	0	删除标志

图 3-62

发送端系统数据如图 3-63 所示。

数据浏览器：表格 ZTTEST_T 选择条目　　18

检查表格...

类	班级	学年	学生番号	语文成绩	数学成绩	自然成绩	思品成绩	创建人	创建日期	创建时间	修改人	修改日期	修改时间	删除标志
000	1A	2009	00001	60	68	70	81			00:00:00			00:00:00	
000	1A	2009	00002	62	65	74	83			00:00:00			00:00:00	
000	1A	2009	00003	62	65	71	89			00:00:00			00:00:00	
000	1A	2009	00004	61	63	75	88			00:00:00			00:00:00	
000	1A	2009	00005	64	66	75	88			00:00:00			00:00:00	
000	1A	2009	00006	64	66	75	88			00:00:00			00:00:00	
000	1A	2009	00007	64	66	76	88			00:00:00			00:00:00	
000	1A	2009	00008	62	66	72	88			00:00:00			00:00:00	
000	1A	2009	00009	62	63	72	82			00:00:00			00:00:00	
000	1A	2009	00010	62	63	74	85			00:00:00			00:00:00	
000	1B	2009	00001	10	10	10	10			00:00:00			00:00:00	
000	1B	2010	00001	30	30	30	30			00:00:00			00:00:00	
000	1B	2010	00002	30	30	30	30			00:00:00			00:00:00	
000	1B	2010	00003	20	20	20	20			00:00:00			00:00:00	
000	1B	2010	00004	50	50	50	50			00:00:00			00:00:00	
000	1B	2010	00005	60	60	60	60			00:00:00			00:00:00	
000	1C	2010	00001	20	20	20	20			00:00:00			00:00:00	
000	1C	2010	00002	70	70	70	70			00:00:00			00:00:00	

图 3-63

接收端系统数据如图 3–64 所示。



图 3–64

2. 定义系统间连接

（1）在接收方系统定义

步骤一：定义发送方与接受方逻辑系统。

按照路径“IDoc 接口 / 启用应用程序链接（ALE）→基本设置→逻辑系统→定义逻辑系统”或运行事务代码 BD54，添加逻辑系统如图 3–65 所示。

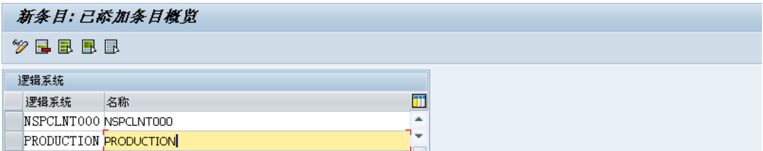


图 3–65

如上图所示，逻辑系统 NSPCLNT000 为接收方用，PRODUCTION 为发送方用。

步骤二：定义客户端与逻辑系统之间的对应关系。

运行事务代码 SCC4，设定客户端信息如图 3–66 所示。

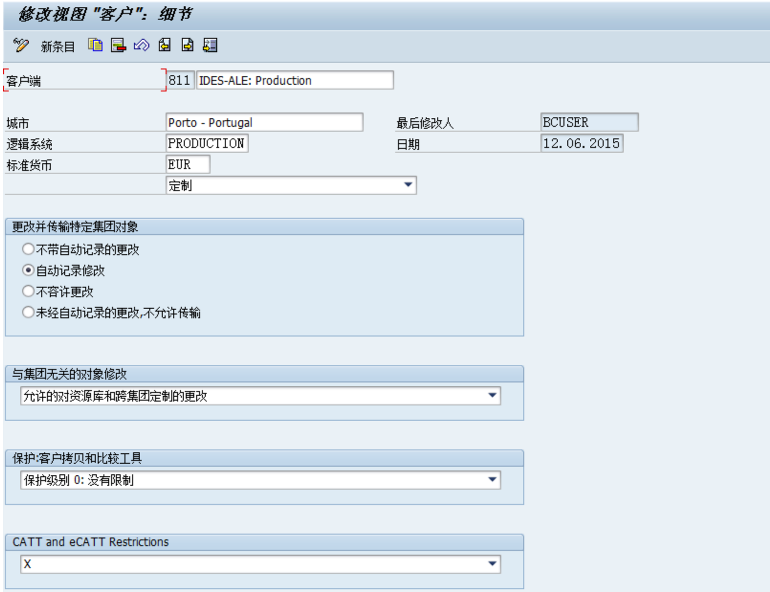


图 3–66

步骤三：定义 RFC 调用目标系统，建立连接。

运行事务代码 SM59，设定 RFC 连接信息如图 3–67 所示。

定义接受方 RFC 连接：填入接收方目标系统名称、IP 地址和实例编号。

图 3–67

输入语言、客户端号、用户名和密码，如图 3–68 所示。

图 3–68

选择“MDMP& Unicode”选项，图 3–69 所示。

RFC终点PRODUCTION

远程登录　连接　Unicode Test

RFC 目标系统　PRODUCTION

连接类型　3　R/3连接　Description

说明

Description 1　XIP集团811

Description 2

Description 3

Administration　Technical Settings　登录　MDMP & Unicode　Special Options

Communication Type with Target System

Non-Unicode

MDMP Settings

Inactive　Active

Unicode

Character Conversion

Default Setting　'#' = U+0023

Short Dump After Conversion Error

Ignore Conversion Errors

Display of Conversion Errors

Error Indicator　#

U+　0023

图 3–69

测试连接和权限测试，单击应用工具栏中的 连接 按钮，如图 3–70 所示。

R F C-连接测验

连接测试 PRODUCTION

连接类型 R/3 连接

Aktion	Ergebnis
注册：	101 msec
Übertragung 0 KB:	1 msec
Übertragung 10 KB:	1 msec
Übertragung 20 KB:	2 msec
Übertragung 30 KB:	2 msec

图 3–70

定义发送方 RFC 连接：填入发送方目标系统名称、IP 地址和实例编号，如图 3–71 所示。

图 3-71

输入语言、客户端号、用户名和密码，如图 3-72 所示。

图 3-72

选择“MDMP& Unicode”选项（该系统为 Non-Unicode 系统），如图 3–73 所示。

图 3–73

测试连接和权限测试，单击应用工具栏中的 连接 按钮，如图 3–74 所示。

图 3–74

（2）在发送方系统定义

按照以上步骤，在发送发定义相同的逻辑系统和对应的 RFC 连接。

3．定义系统间IDoc信息

（1）在接收方系统定义

步骤一：在发送系统中定义数据段。

运行事务代码 WE31，设定数据段信息如图 3–75 所示。



图 3-75

单击新建按钮，填写段中数据项如图 3-76 所示。

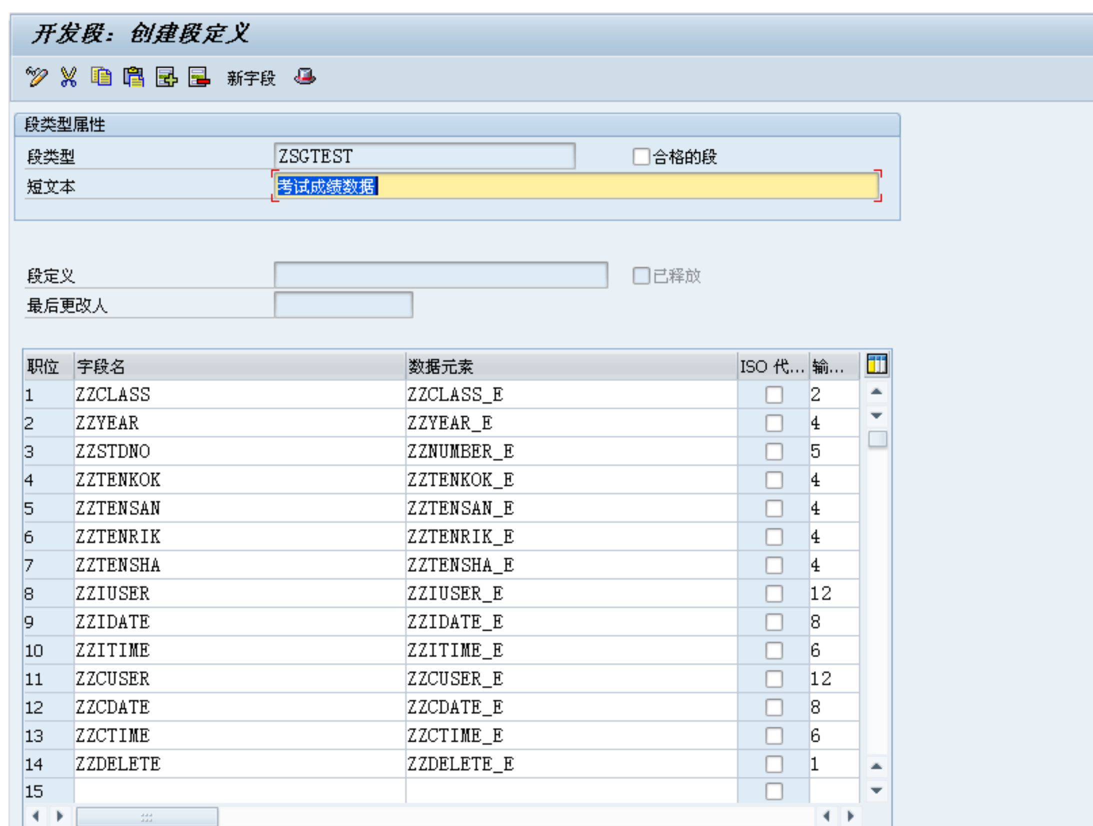


图 3-76

单击标准工具栏中的保存按钮，如图 3-77 所示。

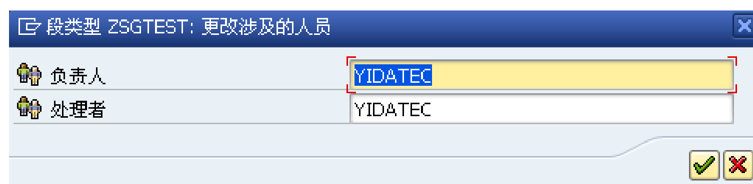


图 3-77

单击上图中的按钮，如图 3-78 所示。

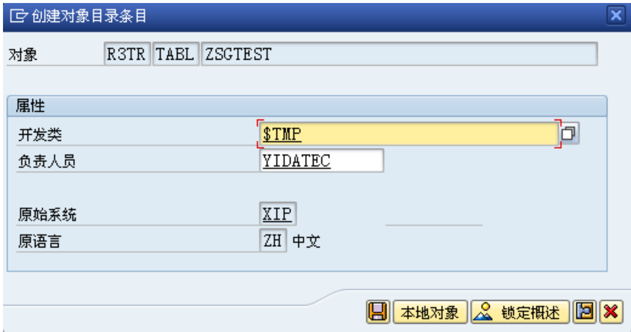


图 3-78

单击上图中的按钮，如图 3-79 所示，其效果如图 3-80 所示。

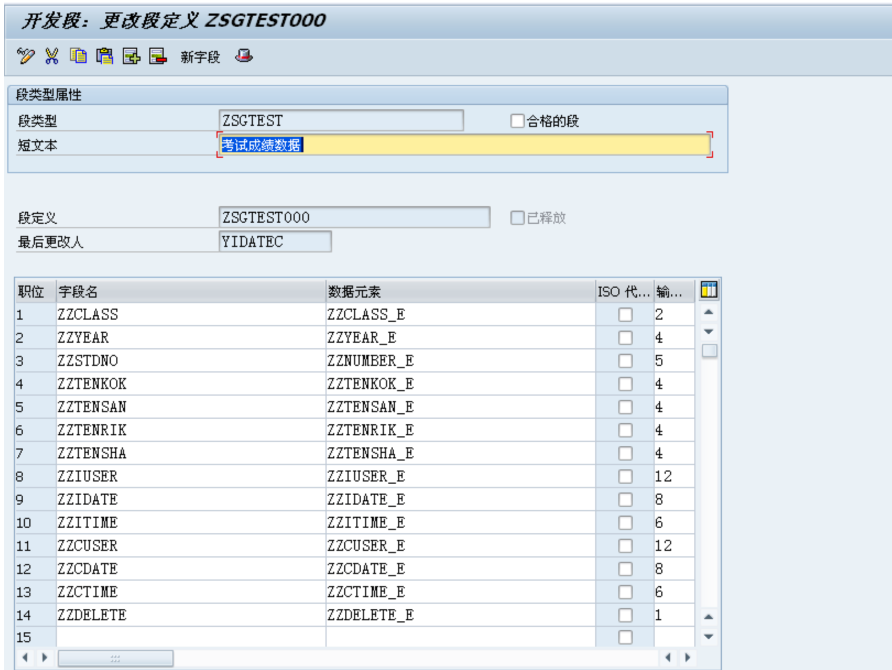


图 3-79



图 3-80

单击标准工具栏上的返回按钮，在段编辑初始画面中按照以下路径发布该段，如图 3-81 所示。

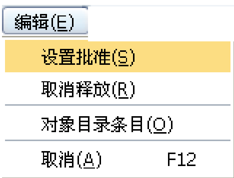


图 3-81

结果如图 3-82 所示。

图 3-82

步骤二：在发送系统中定义基本 IDoc 类型。

运行事务代码 WE30，设定基本 IDoc 类型如图 3-83 所示。

图 3-83

单击新建按钮，填写类型描述如图 3-84 所示。

图 3-84

单击上图中的按钮，如图 3-85 所示。



图 3-85

选中信息类型，单击新建段按钮，填写段及数量编号信息，如图 3-86 所示。



图 3-86

单击上图中的按钮，如图 3-87 所示。

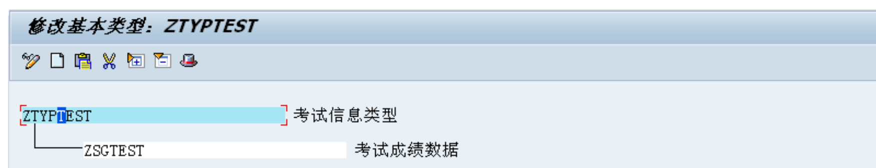


图 3-87

单击标准工具栏中的保存按钮，如图 3-88 所示。

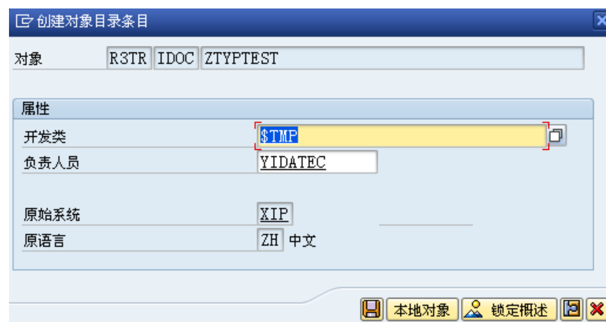


图 3-88

单击上图中的按钮，如图 3-89 所示。



图 3-89

单击标准工具栏中的返回按钮，在段编辑初始画面中按照以下路径发布该类型，如图 3–90 所示。

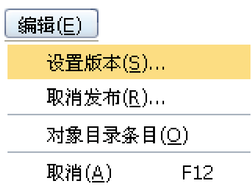


图 3–90

单击图 3–91 中的 是 按钮。

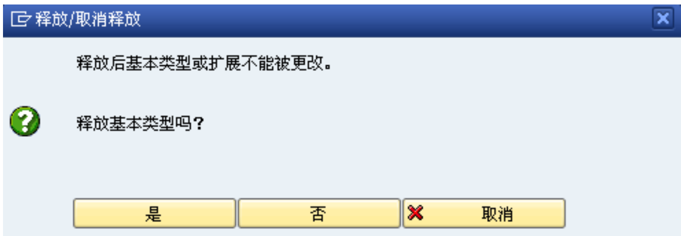


图 3–91

（2）在发送方系统定义

按照以上步骤，在发送方定义相同的段和信息类型。

4．定义系统间IDoc关联

（1）在发送方系统定义

步骤一：创建一个逻辑消息类型。

运行事务代码 WE81，设定数据段信息如图 3–92 所示。



图 3–92

步骤二：指定逻辑消息类型和 IDoc 基本类型的对应关系。

运行事务代码 WE82，设定对应关系如图 3–93 所示。



图 3–93

（2）在接收方系统定义

按照以上步骤，在接收方定义相同的逻辑消息类型和分配。

5．定义系统间IDoc连接

（1）在发送方系统定义

步骤一：创建并分配添加消息类型。

运行事务代码 BD64，创建模型视图，如图 3–94 所示。



图 3–94

单击按钮并改为修改模式，单击创建模型视图按钮创建模型视图，输入描述与 IDoc 名称，如图 3–95 所示。



图 3–95

单击上图中的按钮，如图 3–96 所示。



图 3–96

选中模型视图，单击添加消息类型按钮添加消息模型，输入模型名称、发送方逻辑名、接受方逻辑名和消息类型，如图 3-97 所示。

图 3-97

单击上图中的按钮，并单击标准工具栏中的保存按钮保存设定信息。

步骤二：生成伙伴参数文件。

按照以下路径生成伙伴参数文件，如图 3-98 所示。

图 3-98

输入合作伙伴系统（发送方逻辑系统名称）如图 3-99 所示，单击运行按钮。

图 3-99

执行结果如图 3-100 所示。

生成伙伴参数文件

用于生成伙伴参数文件的日志

伙伴	
系统 NSPCLNT000	伙伴 NSPCLNT000 作为伙伴已经被创建
系统 PRODUCTION	伙伴 PRODUCTION 作为伙伴已经被创建
端口	
系统 PRODUCTION	带有 RFC 目标 A000000036 的端口 PRODUCTION 已经创建
外向参数	
系统 PRODUCTION	消息类型为SYNCH SYNCHRON 的外向参数成功创建 消息类型为ZLTYPTEST ZTYPTEST 的外向参数成功创建

图 3-100

步骤三：分配观察模型。

单击标准工具栏中的返回按钮，按照以下路径将模型分配给逻辑系统，如图 3-101 所示。

编辑(E)
删除(L) Shift+F2
筛选器显示(F) Ctrl+F3
展开子树(T) Ctrl+Shift+F11
折叠子树(U) Ctrl+Shift+F12
模型观察(M)
添加 BAPI Ctrl+F6
添加消息类型(E) Ctrl+F7
显示/编辑明细(I) F2
取消(A) F12
创建(E) Ctrl+F4
使用模板创建(A) Ctrl+F5
复制(C) Ctrl+F12
分配(I)
传送(T)

图 3-101

选择发送方逻辑名称，如图 3-102 所示。

分发模型视图

模型观察 TESTTSIDOC

模块视图接收器

逻辑系统	技术名称
APOCLNT001	APOCLNT001
NSPCLNT001	NSPCLNT001
PRODUCTION	PRODUCTION
local800	T90CLNT090

图 3-102

单击上图中的按钮，分配结果如图 3-103 所示。

模型视图分配的日志

模型视图 TESTTSIDOC 的分配

目标系统 PRODUCTION	模型视图 TESTTSIDOC 已创建

图 3-103

接收方生成结果如图 3-104 所示。

显示分配模型

筛选器模型显示　创建模型视图　添加 BAPI　添加消息类型

分配模型	描述/技术名	业务对象
模型视图		
AL	AL　. 不存在短文本	
ALE-AUDIT	ALE-AUDIT . 不存在短文本	
CONSOLIDAT	CONSOLIDAT. 不存在短文本	
CONTRLDATA	CONTRLDATA. 不存在短文本	
EBP_TO_R3	EBP_TO_R3 . 不存在短文本	
LOG-HR	LOG-HR　. 不存在短文本	
MDM30	MDM30　. 不存在短文本	
MDMDEMO811	MDMDEMO811. 不存在短文本	
MDM_001	MDM_001 . 不存在短文本	
MDM_002	MDM_002 . 不存在短文本	
MDM_003	MDM_003 . 不存在短文本	
MDM_004	MDM_004 . 不存在短文本	
MDM_005	MDM_005 . 不存在短文本	
MDM_006	MDM_006 . 不存在短文本	
MDM_007	MDM_007 . 不存在短文本	
MDM_008	MDM_008 . 不存在短文本	
MDM_009	MDM_009 . 不存在短文本	
MDM_010	MDM_010 . 不存在短文本	
SUBSYSTEMS	SUBSYSTEMS. 不存在短文本	
IDOC Test	TSTMATIDOC	
考试成绩	TESTTSIDOC	
NSPCLNT000	NSPCLNT000	
XIP 集团 811	PRODUCTION	
ZLTYPTEST		

图 3-104

运行事务代码 WE21，确认生成 IDoc 端口，如图 3-105 所示。

IDOC 处理中的端口

端口	描述
端口	
事务性 RFC	
A000000036	PRODUCTION
文件	
CPI-C	
ABAP-PI	
XML 文件	
XML HTTP	

端口 A000000036

描述 PRODUCTION

版本

SAP 版本 3.0/3.1 的 IDoc 记录类型

SAP 4.x 版本的 IDoc 记录类型

RFC 目标 PRODUCTION

图 3-105

运行事务代码 WE20，确认对应合作伙伴参数如图 3–106 所示。



图 3–106

双击消息类型 ZLTYPTEST，确认出站配置，如图 3–107 所示。



图 3–107

步骤四：编辑 IDoc 发布程序。

运行事务 SE38，编写该 IDoc 的出站处理程序，如图 3–108 所示。

```
*-------------------------------------------------------------------*
*
* Program ID              ： ZIDOC_SEND
* Program Description ： 学生成绩发送
* Create  User            ： BCUSER
* Create  Date            ： 2011/02/20
*
*-------------------------------------------------------------------*
* Function Overview
* 程序实现以班级、年度、学生为参照进行交互式查询
* 并将查询结果以 IDoc 形式发送
*
*-------------------------------------------------------------------*

* Change List          :
*      change NO.   Change Date    Change User      Change Detail
*      0000001       2011/02/20      BCUSER          程序创建
*-------------------------------------------------------------------*

REPORT ZIDOC_SEND
  NO STANDARD PAGE HEADING
    LINE-SIZE 120
    LINE-COUNT 90(3)
    MESSAGE-ID ZTESTMSG.

*-------------------------------------------------------------------*
* INCLUDE
*-------------------------------------------------------------------*
INCLUDE: ZTEST0N,         "常量定义
              ZTEST0F.         "Form 定义

*-------------------------------------------------------------------*
* TYPES
*-------------------------------------------------------------------*
*  成绩表信息
TYPES: BEGIN    OF   TYP_TEST.
             INCLUDE    STRUCTURE   ZTTEST_T.
TYPES:                  END       OF   TYP_TEST.
*  成绩表
TYPES: BEGIN OF TYP_TESTLIST,
             ZZCLASS       LIKE   ZTTEST_T-ZZCLASS,               "班级
             ZZYEAR        LIKE   ZTTEST_T-ZZYEAR,                "学年
             ZZSTDNO       LIKE   ZTTEST_T-ZZSTDNO,               "学号
             ZZTENKOK    LIKE   ZTTEST_T-ZZTENKOK,              "语文
             ZZTENSAN     LIKE   ZTTEST_T-ZZTENSAN,              "数学
             ZZTENRIK      LIKE   ZTTEST_T-ZZTENRIK,              "自然
             ZZTENSHA     LIKE   ZTTEST_T-ZZTENSHA,              "思想品德
```

```
      ZZNAME      LIKE   ZMNUMBER_T-ZZNAME,         "姓名
    END     OF   TYP_TESTLIST.

*---------------------------------------------------------------------*
* TABLES:
*---------------------------------------------------------------------*
TABLES: ZTTEST_T.              "考试成绩表

*---------------------------------------------------------------------*
* CONSTANTS
*---------------------------------------------------------------------*
CONSTANTS: LINE88 TYPE I VALUE 88.

*---------------------------------------------------------------------*
* DATA
*---------------------------------------------------------------------*

* 报表处理用变量
DATA: BEGIN OF TNAME,
        NAME1   LIKE   ZMTNUMBER_T-ZTNAME,        "班主任
        NAME2   LIKE   ZMTNUMBER_T-ZTNAME,        "副班主任
      END OF TNAME.
DATA: BEGIN OF FLG,
        CLS_FRST   TYPE   C   VALUE   CNS-OFF,       "班级起始
      END OF FLG.
* 考试成绩表
DATA:  TBL_TEST     TYPE STANDARD TABLE OF TYP_TESTLIST WITH HEADER LINE.
DATA:  ITEM_LINE LIKE LINE OF TBL_TEST.
DATA:  IDOC_TEST     TYPE STANDARD TABLE OF TYP_TEST WITH HEADER LINE.
* IDoc 应用
DATA: IDOC_CTL LIKE EDIDC,
      SGTEST LIKE ZSGTEST.

DATA: TBL_RET_CTL LIKE EDIDC OCCURS 0 WITH HEADER LINE,
      TBL_IDOC_DATA LIKE EDIDD OCCURS 0 WITH HEADER LINE.

*---------------------------------------------------------------------*
* PARAMETERS
*---------------------------------------------------------------------*
PARAMETERS: P_CLASS          TYPE ZTTEST_T-ZZCLASS   "班级
                             OBLIGATORY .              "必须

PARAMETERS P_YEAR TYPE ZTTEST_T-ZZYEAR OBLIGATORY.  "学年
*---------------------------------------------------------------------*
* INITIALIZATION
*---------------------------------------------------------------------*
```

```
INITIALIZATION.

* 数据初始化
  PERFORM DATA_INIT
                    USING
                         P_CLASS   "班级
                         P_YEAR.    "学年

*-----------------------------------------------------------------------*
* AT SELECTION SCREEN
*-----------------------------------------------------------------------*
AT SELECTION-SCREEN.

* 选择画面项目检查-班级
  PERFORM CLASS_INPUT_CHECK USING P_CLASS TEXT-001 CHANGING TNAME.

* 选择画面项目检查-学年
  PERFORM YEAR_INPUT_CHECK USING P_YEAR TEXT-002.

*-----------------------------------------------------------------------*
*START-OF-SELECTION
*-----------------------------------------------------------------------*
START-OF-SELECTION.

* 报表数据初始化
  PERFORM REPORT_DATA_INIT.
* 报表数据编辑
  PERFORM REPORT_DATA_EDIT.
* 报表数据发送
  PERFORM REPORT_DATA_SEND.
* 报表数据打印
  PERFORM REPORT_DATA_EXE.
*-----------------------------------------------------------------------*
* END-OF-SELECTION
*-----------------------------------------------------------------------*
END-OF-SELECTION.

* 报表程序终了处理
  PERFORM REPORT_EDIT_END.
*-----------------------------------------------------------------------*
* TOP-OF-PAGE
*-----------------------------------------------------------------------*
TOP-OF-PAGE.

* 报表页眉编辑
  PERFORM REPORT_HEADER_EDIT.
```

```
*-------------------------------------------------------------------*
* END-OF-PAGE
*-------------------------------------------------------------------*
END-OF-PAGE.

* 报表页脚编辑
  PERFORM REPORT_FOOTER_EDIT.



*&-------------------------------------------------------------------*
*&      Form  REPORT_DATA_INIT
*&-------------------------------------------------------------------*
*       报表数据初始化
*-------------------------------------------------------------------*
FORM REPORT_DATA_INIT.

* 内部表初始化
   CLEAR:       TBL_TEST.                  "成绩表头
   REFRESH:   TBL_TEST.                   "成绩表
   CLEAR:       IDOC_TEST.                "成绩表头
   REFRESH:   IDOC_TEST.                 "成绩表

ENDFORM.                    " REPORT_DATA_INIT
*&-------------------------------------------------------------------*
*&      Form  REPORT_DATA_EDIT
*&-------------------------------------------------------------------*
*       报表数据编辑
*-------------------------------------------------------------------*
FORM REPORT_DATA_EDIT .
* 成绩表数据抽取
  SELECT      ZZCLASS              "班级
              ZZYEAR               "学年
              ZZSTDNO              "学号
              ZZTENKOK             "语文
              ZZTENSAN             "数学
              ZZTENRIK             "自然
              ZZTENSHA             "思品
              ZZNAME               "姓名
    FROM ZZTEST_V                  "成绩表
    INTO TABLE TBL_TEST            "成绩内部表
    WHERE ZZCLASS  =  P_CLASS   "班级
    AND   ZZYEAR = P_YEAR .     "学年

* 无数据情况
  IF SY-SUBRC <> 0.
     MESSAGE S002.
```

```
  STOP.
 ELSE.
* 成绩表数据抽取
  SELECT    *
   FROM ZTTEST_T            "成绩表
   INTO TABLE IDOC_TEST       "成绩内部表
   WHERE ZZCLASS  =  P_CLASS   "班级
   AND   ZZYEAR = P_YEAR   .   "学年

 ENDIF.

* 排序
 SORT TBL_TEST BY ZZCLASS ASCENDING          "班级
                  ZZYEAR ASCENDING           "学年
                  ZZSTDNO ASCENDING.         "学号
ENDFORM.                 " REPORT_DATA_EDIT
*&---------------------------------------------------------------------*
*&     Form  REPORT_DATA_SEND
*&---------------------------------------------------------------------*
*      报表数据发送
*----------------------------------------------------------------------*
FORM REPORT_DATA_SEND .
*  编辑 IDoc 控制字段
 PERFORM BUILD_IDOC_CONTROL.
*  编辑 IDoc 段内容
 LOOP AT IDOC_TEST.
  PERFORM BUILD_IDOC_SEGMENT.
 ENDLOOP.

*  创建 IDoc 并发送
 PERFORM CREATE_IDOC_SEND.

ENDFORM.                 " REPORT_DATA_SEND
*&---------------------------------------------------------------------*
*&     Form  REPORT_DATA_EXE
*&---------------------------------------------------------------------*
*      报表数据打印
*----------------------------------------------------------------------*
FORM REPORT_DATA_EXE .
 CHECK TBL_TEST[] IS NOT INITIAL.

 LOOP AT TBL_TEST.                     "成绩内部表
  CLEAR FLG-CLS_FRST.                  "班级表示清空
  AT NEW ZZCLASS.                      "新班级
   MOVE CNS-ON TO FLG-CLS_FRST.        "班级标示设 ON
  ENDAT .
```

```
    RESERVE 3  LINES.
*   报表明细编辑
    PERFORM EDIT_REPORT_DTL.

    AT END OF ZZCLASS .              "班级最后
*    报表挂线
      PERFORM EDIT_REPORT_CLS.
    ENDAT.

    AT END OF ZZYEAR.
      NEW-PAGE.
    ENDAT.

  ENDLOOP.

  LOOP AT TBL_RET_CTL.
    WRITE:/ 'IDoc Generated - ', TBL_RET_CTL-DOCNUM.
  ENDLOOP.

ENDFORM.                    " REPORT_DATA_EXE
*&---------------------------------------------------------------------*
*&      Form  REPORT_EDIT_END
*&---------------------------------------------------------------------*
*       报表程序终了处理
*----------------------------------------------------------------------*
FORM REPORT_EDIT_END .

  CHECK TBL_TEST[] IS NOT INITIAL.
* 当前行号小于 88 时
  IF SY-LINNO < LINE88 .
    SKIP TO LINE LINE88 .
    PERFORM REPORT_FOOTER_EDIT.
  ENDIF.

ENDFORM.                    " REPORT_EDIT_END
*&---------------------------------------------------------------------*
*&      Form  REPORT_HEADER_EDIT
*&---------------------------------------------------------------------*
*       报表页眉编辑
*----------------------------------------------------------------------*
FORM REPORT_HEADER_EDIT .
  SKIP 1 .
  WRITE:/030 TEXT-003.               "成绩表
  WRITE:/016 TEXT-002 ,              "学年
              020 CNS-COLON,
              022 TBL_TEST-ZZYEAR.
```

```
 WRITE:  030  TEXT-004,            "班主任
          036 CNS-COLON,
          038 TNAME-NAME1(6),
          045 TNAME-NAME2(6).
 WRITE:/015 SY-ULINE(052) .
 WRITE:/015 SY-VLINE,
       017 TEXT-001,              "班级
       021 SY-VLINE,
       025 TEXT-005,              "学生
       038 SY-VLINE,
       040 TEXT-006,              "语文
       045 SY-VLINE,
       047 TEXT-007,             "数学
       052 SY-VLINE,
       054 TEXT-008,             "自然
       059 SY-VLINE,
       061 TEXT-009,             "思品
       066 SY-VLINE.
 WRITE:/015 SY-VLINE,
       021 SY-VLINE,
       038 SY-VLINE,
       045 SY-VLINE,
       052 SY-VLINE,
       059 SY-VLINE,
       066 SY-VLINE,
       015 SY-ULINE(052).

ENDFORM.                  " REPORT_HEADER_EDIT
*&---------------------------------------------------------------------*
*&      Form  EDIT_REPORT_DTL
*&---------------------------------------------------------------------*
*       报表明细编辑
*----------------------------------------------------------------------*
FORM EDIT_REPORT_DTL .

 WRITE:/015    SY-VLINE,
        021    SY-VLINE,
        023    TBL_TEST-ZZNAME(14),    "学生
        038    SY-VLINE,
        041(4) TBL_TEST-ZZTENKOK,      "语文
        045    SY-VLINE,
        048(4) TBL_TEST-ZZTENSAN,      "数学
        052    SY-VLINE,
        055(4) TBL_TEST-ZZTENRIK,      "自然
        059    SY-VLINE,
        062(4) TBL_TEST-ZZTENSHA,      "思品
```

```
        066   SY-VLINE.
  IF FLG-CLS_FRST  = CNS-ON.          "班级开始
   WRITE: 017 TBL_TEST-ZZCLASS+0(1),   "
        018 CNS-PLUS ,
        019 TBL_TEST-ZZCLASS+1(1).
  ENDIF.

ENDFORM.                    " EDIT_REPORT_DTL
*&---------------------------------------------------------------------*
*&      Form  EDIT_REPORT_CLS
*&---------------------------------------------------------------------*
*       报表挂线
*----------------------------------------------------------------------*
FORM EDIT_REPORT_CLS .

* 成绩表挂线
  WRITE:/015 SY-VLINE,
       021 SY-VLINE,
       038 SY-VLINE,
       045 SY-VLINE,
       052 SY-VLINE,
       059 SY-VLINE,
       066 SY-VLINE,
       015 SY-ULINE(052).

ENDFORM.                    " EDIT_REPORT_CLS
*&---------------------------------------------------------------------*
*&      Form  REPORT_FOOTER_EDIT
*&---------------------------------------------------------------------*
*       报表页脚编辑
*----------------------------------------------------------------------*
FORM REPORT_FOOTER_EDIT .

  WRITE 030 SY-PAGNO.       "页数

ENDFORM.                    " REPORT_FOOTER_EDIT
*&---------------------------------------------------------------------*
*&      Form  BUILD_IDOC_CONTROL
*&---------------------------------------------------------------------*
*       编辑 IDoc 控制字段
*----------------------------------------------------------------------*
FORM BUILD_IDOC_CONTROL .
* 类型
  IDOC_CTL-IDOCTP = 'ZTYPTEST'.    "IDoc 类型
  IDOC_CTL-MESTYP = 'ZLTYPTEST'.     "信息类型
```

```
* 接收端
 IDOC_CTL-RCVPRT = 'LS'.        "伙伴类型
 IDOC_CTL-RCVPRN = 'PRODUCTION'.  "合作伙伴编号
 IDOC_CTL-RCVPOR = 'A000000036'.   "端口
* 发送端
 IDOC_CTL-SNDPRT = 'LS'.         "伙伴类型
 IDOC_CTL-SNDPRN = 'NSPCLNT000'.   "合作伙伴编号

ENDFORM.                  " BUILD_IDOC_CONTROL
*&---------------------------------------------------------------------*
*&      Form  BUILD_IDOC_SEGMENT
*&---------------------------------------------------------------------*
*       编辑 IDoc 段内容
*----------------------------------------------------------------------*
FORM BUILD_IDOC_SEGMENT .
* 表头信息
 TBL_IDOC_DATA-SEGNAM = 'ZSGTEST'. "段名
 TBL_IDOC_DATA-SEGNUM = SY-TABIX.           "段 No
 MOVE-CORRESPONDING IDOC_TEST TO SGTEST.
 MOVE SGTEST TO TBL_IDOC_DATA-SDATA.
 APPEND TBL_IDOC_DATA.

ENDFORM.                  " BUILD_IDOC_SEGMENT
*&---------------------------------------------------------------------*
*&      Form  CREATE_IDOC_SEND
*&---------------------------------------------------------------------*
*       创建 IDoc 并发送
*----------------------------------------------------------------------*
FORM CREATE_IDOC_SEND .
  CALL FUNCTION 'MASTER_IDOC_DISTRIBUTE'
    EXPORTING
      MASTER_IDOC_CONTROL                  = IDOC_CTL
    TABLES
      COMMUNICATION_IDOC_CONTROL           = TBL_RET_CTL
      MASTER_IDOC_DATA                     = TBL_IDOC_DATA
    EXCEPTIONS
      ERROR_IN_IDOC_CONTROL                = 1
      ERROR_WRITING_IDOC_STATUS            = 2
      ERROR_IN_IDOC_DATA                   = 3
      SENDING_LOGICAL_SYSTEM_UNKNOWN       = 4
      OTHERS                               = 5.

 IF SY-SUBRC <> 0.
   MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
     WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
```

```
      ELSE.
        COMMIT WORK.
      ENDIF.

    ENDFORM.                    " CREATE_IDOC_SEND
```

文本元素编辑如图 3-108 所示。

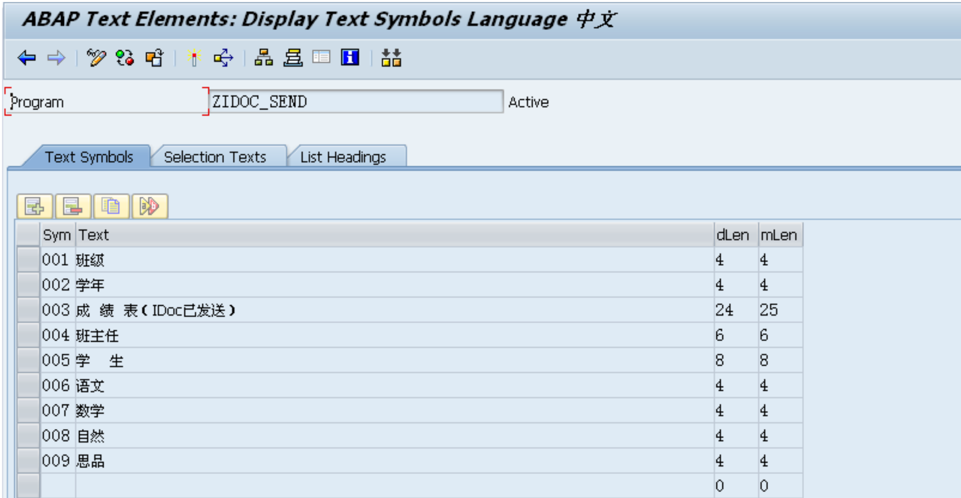


图 3-108

选择文本元素编辑如图 3-109 所示。

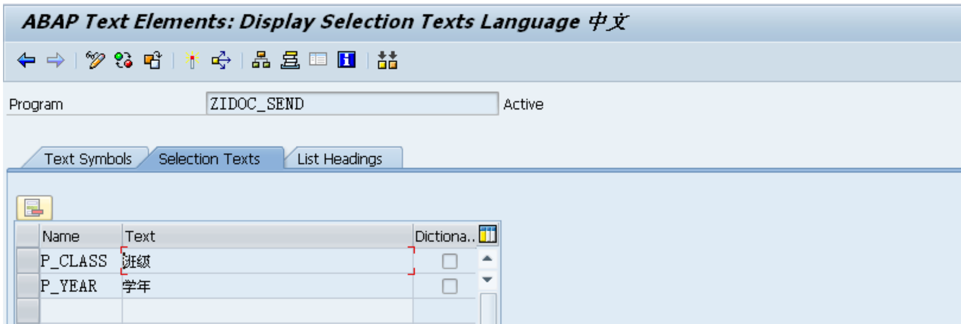


图 3-109

（2）在接收方系统定义

步骤一：生成伙伴参数文件。

运行事务代码 BD64，选中已生成的模型，按照以下路径生成伙伴参数文件，如图 3-110 所示。

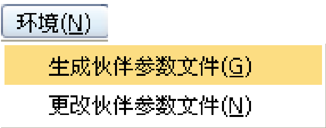


图 3-110

输入合作伙伴系统（接收方逻辑系统名称），如图 3-111 所示，单击运行按钮。

生成伙伴参数文件

模型视图 TESTTSIDOC 到
合作伙伴系统 NSPCLNT000 到
检查运行

合作伙伴参数文件的缺省参数
后处理：已授权的用户
类型 US 用户
ID YIDATEC YIDATEC
外向参数
版本 3 版本 4.0 以上的凭证记录类型
包大小 100 IDocs
输出模式
立即转换 IDoc
收集 IDocs 并转换
内向参数
处理
立即触发
由后台程序触发

图 3-111

显示成功信息如图 3-112 所示。

生成伙伴参数文件

用于生成伙伴参数文件的日志

伙伴	
系统 NSPCLNT000	伙伴 NSPCLNT000 作为伙伴已经被创建
系统 PRODUCTION	系统 PRODUCTION 作为伙伴类型已经存在
端口	
系统 NSPCLNT000	带有 RFC 目标 A000000017 的端口 NSPCLNT000 已经创建
外向参数	
系统 NSPCLNT000	消息类型为SYNCH SYNCHRON 的外向参数成功创建

图 3-112

运行事务 WE21 确认端口信息，如图 3-113 所示。

IDOC 处理中的端口

端口	描述
端口	
事务性 RFC	
A000000002	Sales system
A000000003	Production system
A000000004	Central system
A000000006	ID3 IDES SENDER 802
A000000007	BC_CONSOLIDATOR
A000000008	ID3 IDES 800
A000000009	ID3-AM
A000000010	Q30CLNT002
A000000011	Port to ICARO_IZY

端口 A000000017
描述 NSPCLNT000
版本
SAP 版本 3.0/3.1 的 IDoc 记录类型
SAP 4.x 版本的 IDoc 记录类型
RFC 目标 NSPCLNT000

图 3-113

步骤二：创建入站处理函数。

运行事务代码 SE37，单击拷贝按钮，如图 3-114 所示。

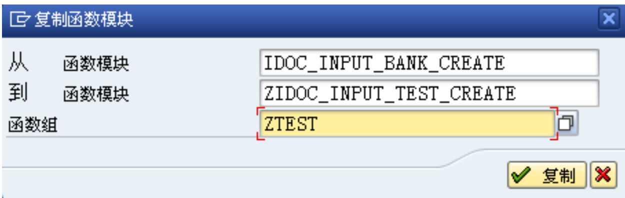


图 3-114

注：一般由已存在的入站函数拷贝得来，可以提高效率。

单击上图中的 复制 按钮，效果如图 3-115 所示。



图 3-115

编辑函数 ZIDOC_INPUT_TEST_CREATE 如下：

编辑属性标签页，如图 3-116 所示。

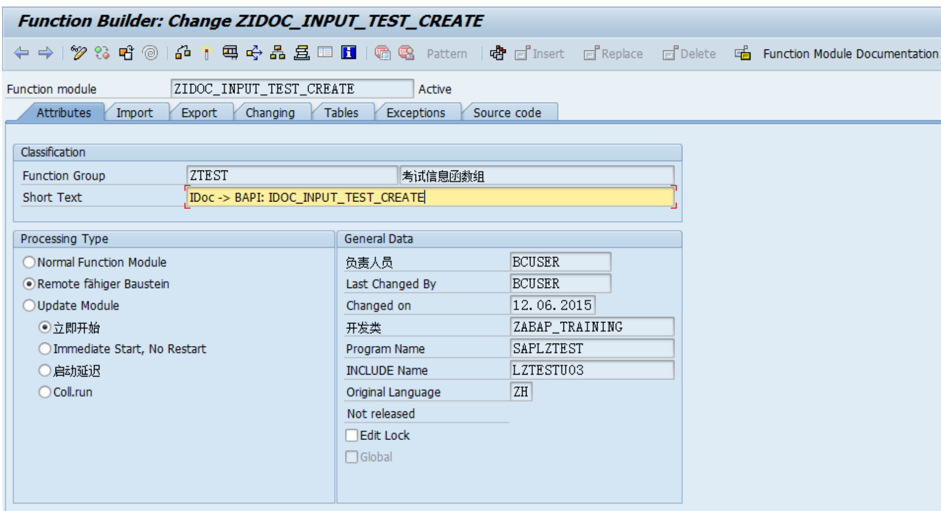


图 3-116

编辑输入标签页，如图 3-117 所示。

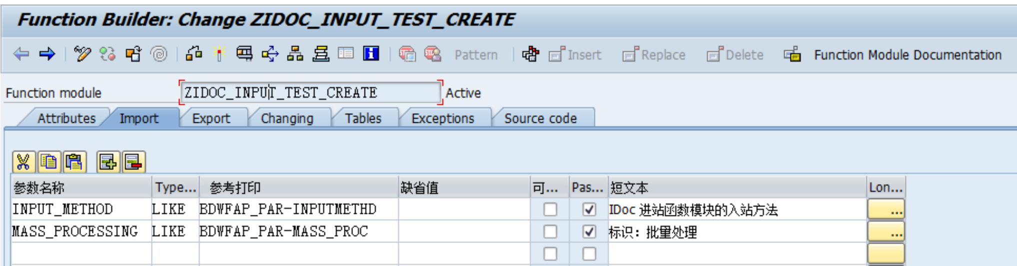


图 3-117

编辑“Export”标签页，如图 3-118 所示。

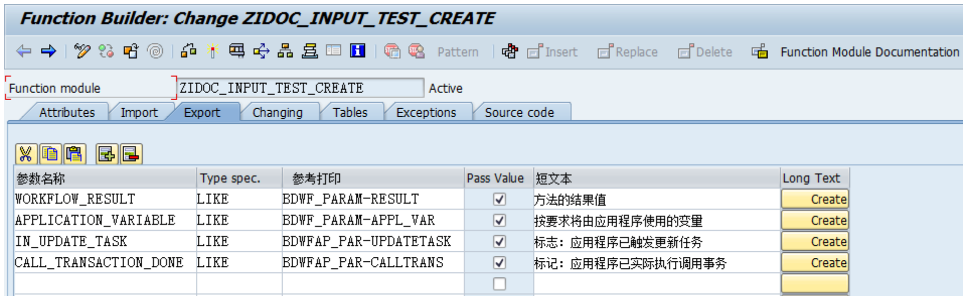


图 3-118

编辑“Changing”标签页，如图 3-119 所示。

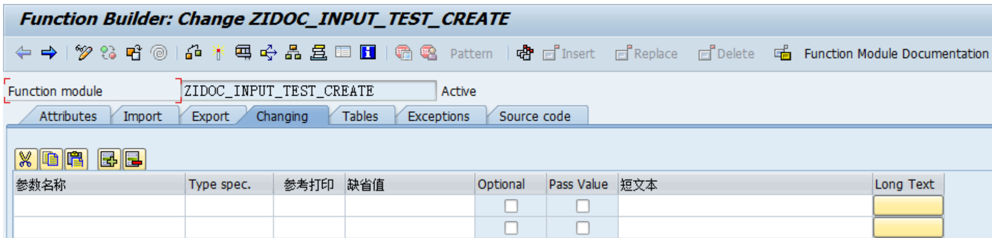


图 3-119

编辑“Tables”标签页，如图 3-120 所示。

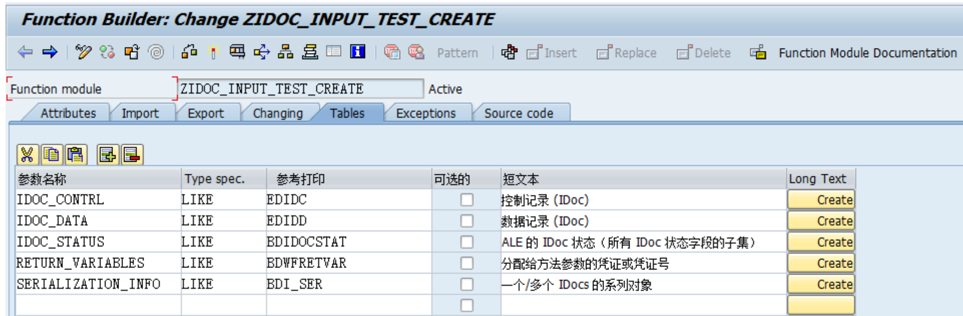


图 3-120

编辑“Exceptions”标签页，如图 3-121 所示。

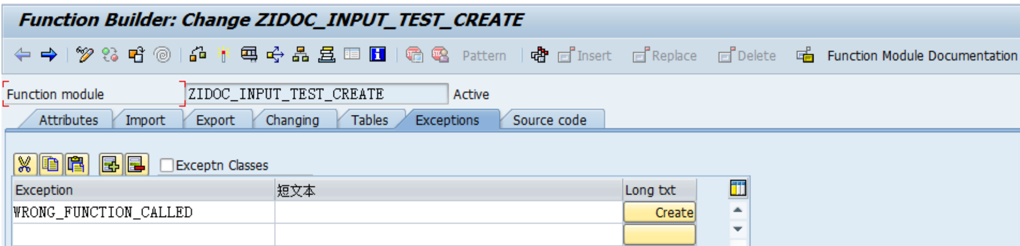


图 3-121

源码如下：

```
FUNCTION ZIDOC_INPUT_TEST_CREATE.
```

```
*"----------------------------------------------------------------------
*"*"Local interface:
*"  IMPORTING
*"     VALUE(INPUT_METHOD) LIKE  BDWFAP_PAR-INPUTMETHD
*"     VALUE(MASS_PROCESSING) LIKE  BDWFAP_PAR-MASS_PROC
*"  EXPORTING
*"     VALUE(WORKFLOW_RESULT) LIKE  BDWF_PARAM-RESULT
*"     VALUE(APPLICATION_VARIABLE) LIKE  BDWF_PARAM-APPL_VAR
*"     VALUE(IN_UPDATE_TASK) LIKE  BDWFAP_PAR-UPDATETASK
*"     VALUE(CALL_TRANSACTION_DONE) LIKE  BDWFAP_PAR-CALLTRANS
*"  TABLES
*"      IDOC_CONTRL STRUCTURE  EDIDC
*"      IDOC_DATA STRUCTURE  EDIDD
*"      IDOC_STATUS STRUCTURE  BDIDOCSTAT
*"      RETURN_VARIABLES STRUCTURE  BDWFRETVAR
*"      SERIALIZATION_INFO STRUCTURE  BDI_SER
*"  EXCEPTIONS
*"      WRONG_FUNCTION_CALLED
*"----------------------------------------------------------------------

* 声明常量
  INCLUDE MBDCONWF.
* 声明表结构
  TABLES : ZTTEST_T.
* 声明段结构
  DATA : SGTEST LIKE ZSGTEST.
* 声明操作结果
  DATA WK_SUBRC LIKE SY-SUBRC VALUE 0.

  WORKFLOW_RESULT = C_WF_RESULT_OK.

  LOOP AT IDOC_CONTRL.
*   判断信息类型
    IF IDOC_CONTRL-MESTYP NE 'ZLTYPTEST'.
      RAISE WRONG_FUNCTION_CALLED.
    ENDIF.

*   读取 IDoc 数据
    LOOP  AT IDOC_DATA .
*     赋值数据
      SGTEST = IDOC_DATA-SDATA.
      MOVE-CORRESPONDING SGTEST TO ZTTEST_T.
*     更新数据库
      INSERT INTO ZTTEST_T VALUES ZTTEST_T.
*      失败的场合
      IF SY-SUBRC <> 0.
        WK_SUBRC = 1.
      ENDIF.
```

```
    ENDLOOP.
*   IDoc 处理成功
    IF WK_SUBRC EQ 0.
     IDOC_STATUS-DOCNUM = IDOC_CONTRL-DOCNUM.
     IDOC_STATUS-STATUS = '53'.
     IDOC_STATUS-MSGTY = 'I'.
     IDOC_STATUS-MSGID = 'ZTESTMSG'.
     IDOC_STATUS-MSGNO = '006'.

     APPEND IDOC_STATUS.
     CLEAR IDOC_STATUS.
*   IDoc 处理失败
    ELSE.
     IDOC_STATUS-DOCNUM = IDOC_CONTRL-DOCNUM.
     IDOC_STATUS-STATUS = '51'.
     IDOC_STATUS-MSGTY = 'E'.
     IDOC_STATUS-MSGID = 'ZTESTMSG'.
     IDOC_STATUS-MSGNO = '005'.
     IDOC_STATUS-MSGV1 = SGTEST-ZZCLASS.
     IDOC_STATUS-MSGV2 = SGTEST-ZZYEAR.
     APPEND IDOC_STATUS.
     CLEAR IDOC_STATUS.
     WORKFLOW_RESULT = C_WF_RESULT_ERROR.
     RETURN_VARIABLES-WF_PARAM = 'Error_Idocs'.
     RETURN_VARIABLES-DOC_NUMBER = IDOC_CONTRL-DOCNUM.
     APPEND RETURN_VARIABLES.
     CLEAR RETURN_VARIABLES.
    ENDIF.

   ENDLOOP.

ENDFUNCTION.
```

关联信息对象（事务码 SE91），如图 3-122 所示。

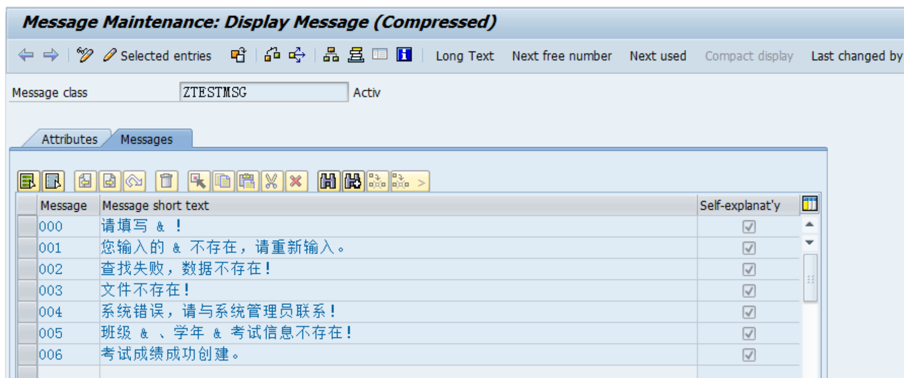


图 3-122

步骤三：将函数分配给逻辑信息。

运行事务 WE57，单击标准工具栏中的按钮，转换视图为变更模式，如图 3–123 所示。



图 3–123

单击 新条目 按钮，新建条目如图 3–124 所示。

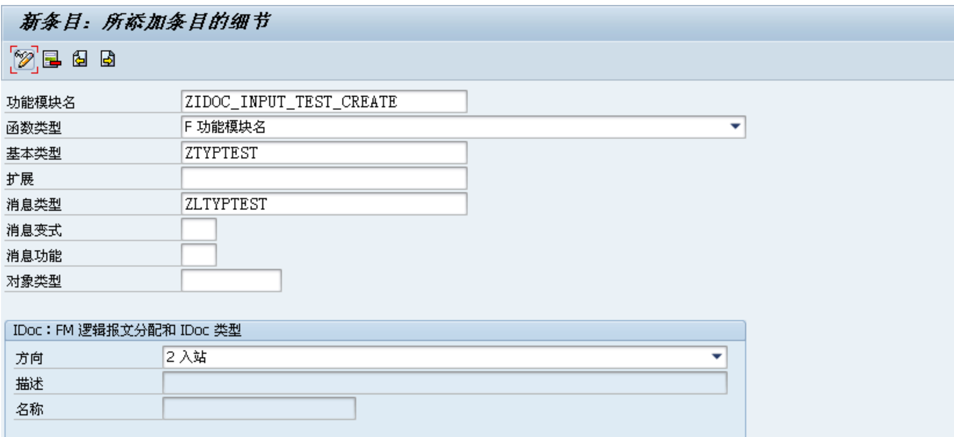


图 3–124

单击标准工具栏中的保存按钮，如图 3–125 所示。



图 3–125

单击上图中的按钮，将分配结果保存。

步骤四：将函数定义为入站函数。

运行事务 BD51，单击标准工具栏中的按钮，转换视图为变更模式，如图 3–126 所示。

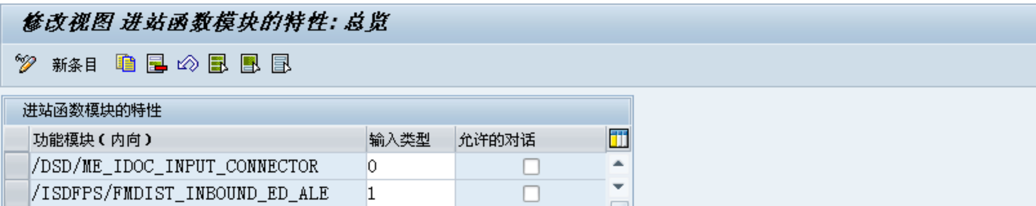


图 3–126

单击 新条目 按钮，新建入站函数如图 3-127 所示。

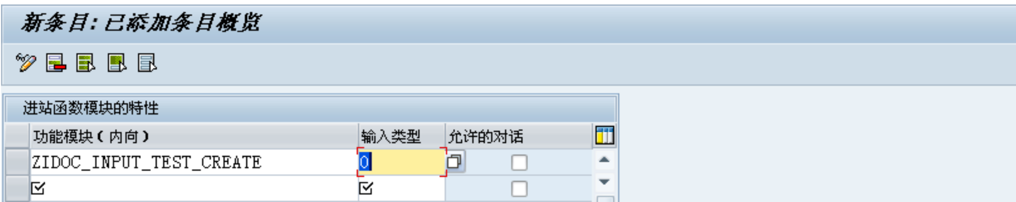


图 3-127

单击标准工具栏中的保存按钮，如图 3-128 所示。



图 3-128

单击上图中的按钮，将添加结果保存。

步骤五：创建该 IDoc 的入站处理代码。

运行事务 WE42，单击标准工具栏中的按钮，转换视图为变更模式，如图 3-129 所示。

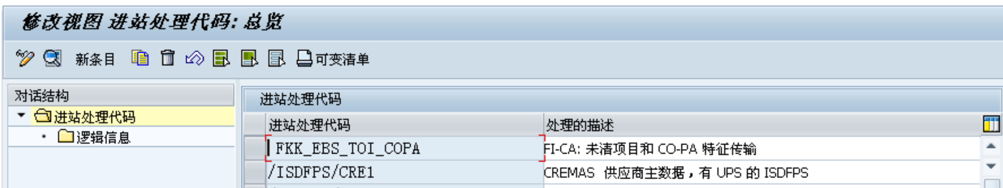


图 3-129

单击 新条目 按钮，新建入站过程代码如图 3-130 所示。



图 3-130

单击标准工具栏中的保存按钮，如图 3-131 所示。



图 3-131

单击上图中的按钮，输入入站函数，如图 3-132 所示。

新条目：所添加条目的细节

处理代码 TSTIDOCIN

模块（内向）

功能模块 ZIDOC_INPUT_TEST_CREATE

重复最大数

IDoc 包

对象类型

结束事件

IDoc

对象类型

开始事件

结束事件

成功事件

应用程序对象

对象类型

开始事件

图 3-132

单击标准工具栏中的保存按钮，如图 3-133 所示。

Prompt for Customizing request

View Cluster Maintenance: VED_TMSG2

号 XIPK900007 定制请求

Short Description IDOC设定

自己的请求

图 3-133

单击上图中的按钮，保存后单击标准工具栏中的返回按钮，如图 3-134 所示。



图 3-134

双击上图中的树形菜单逻辑信息，添加消息类型。单击新条目按钮，新建条目如图 3-135 所示。

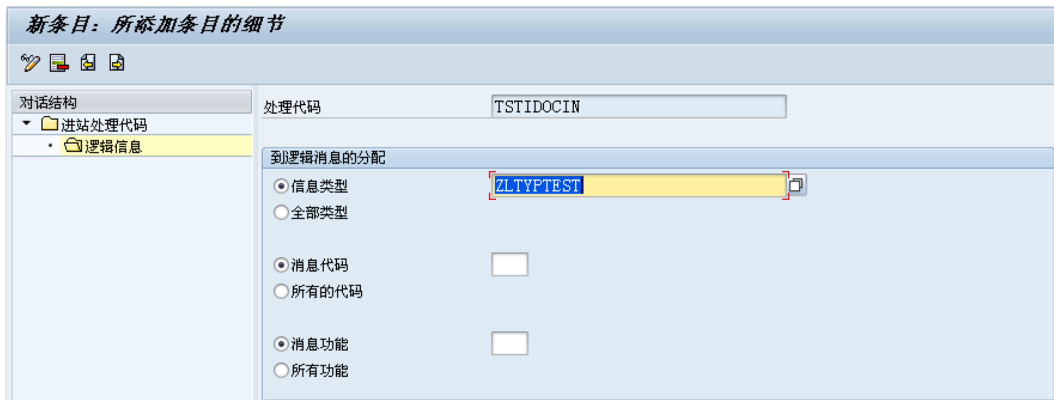


图 3-135

单击标准工具栏上的保存按钮。

步骤六：生成新的伙伴参数文件。

运行事务代码 BD64，选中已生成的模型，按照以下路径生成伙伴参数文件，如图 3-136 所示。

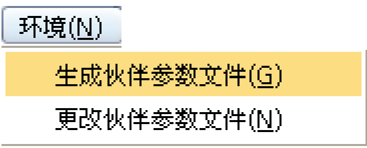


图 3-136

输入合作伙伴系统（接收方逻辑系统名称），如图 3-137 所示，单击运行按钮。

生成伙伴参数文件

模型视图	TESTTSIDOC	到	
合作伙伴系统	NSPCLNT000	到	
检查运行			

合作伙伴参数文件的缺省参数

后处理：已授权的用户

类型	US 用户
ID	YIDATEC YIDATEC

外向参数

版本	3 版本 4.0 以上的凭证记录类型
包大小	100 IDocs

输出模式

- 立即转换 IDoc
- 收集 IDocs 并转换

内向参数

处理

- 立即触发
- 由后台程序触发

图 3-137

显示成功信息如图 3-138 所示。

生成伙伴参数文件

用于生成伙伴参数文件的日志	
伙伴	
系统 NSPCLNT000	系统 NSPCLNT000 作为伙伴类型已经存在
系统 PRODUCTION	系统 PRODUCTION 作为伙伴类型已经存在
端口	
系统 NSPCLNT000	带有 RFC 目标 A000000017 的端口 NSPCLNT000 已经存在
外向参数	
系统 NSPCLNT000	消息类型为SYNCH SYNCHRON 的外向参数已经存在
内向参数	
系统 NSPCLNT000	消息类型为 ZLTYPTEST 的内向参数已成功创建

图 3-138

6．测试系统间IDoc连接

（1）在发送端发送 IDoc 信息

步骤一：运行程序 ZIDOC_SEND，运行结果如图 3-139 所示。

发送考试信息

班级	1A
学年	2009

图 3-139

单击运行按钮，效果如图 3-140 所示。

发送考试信息

成　绩　表（IDoc已发送）

学年：2009　　班主任：刘备　　严长寿

班级	学　　生	语文	数学	自然	思品
1-A	郑喜定	60	68	70	81
	魏喜奎	62	65	74	83
	金喜善	62	65	71	89
	洪喜官	61	63	75	88
	朴喜珍	64	66	75	88
	王喜凤	64	66	75	88
	刘喜仁	64	66	76	88
	白喜医	62	66	72	88
	赵喜庆	62	63	72	82
	张喜早	62	63	74	85

图 3-140

步骤二：查看发送成绩结果。

发送完成后，运行事务 WE02 查看发送消息，如图 3-141 所示。

图 3-141

单击运行按钮，如图 3-142 所示。

图 3-142

（2）在接收端接收

步骤一：查看接收消息。

在接受方服务器运行事务 WE02，查看接受消息，如图 3-143 所示。



图 3-143

步骤二：查看接收成绩。

在接受方服务器运行事务 SE16，查看表信息，如图 3-144 所示。

数据浏览器：表格 ZTTEST_T 选择条目 10

检查表格...

类	班级	学年	学生番号	语文成绩	数学成绩	自然成绩	思品成绩	创建人	创建日期	创建时间	修改人	修改日期	修改时间	删除标志
811	1A	2009	00001	60	68	70	81			00:00:00			00:00:00	
811	1A	2009	00002	62	65	74	83			00:00:00			00:00:00	
811	1A	2009	00003	62	65	71	89			00:00:00			00:00:00	
811	1A	2009	00004	61	63	75	88			00:00:00			00:00:00	
811	1A	2009	00005	64	66	75	88			00:00:00			00:00:00	
811	1A	2009	00006	64	66	75	88			00:00:00			00:00:00	
811	1A	2009	00007	64	66	76	88			00:00:00			00:00:00	
811	1A	2009	00008	62	66	72	88			00:00:00			00:00:00	
811	1A	2009	00009	62	63	72	82			00:00:00			00:00:00	
811	1A	2009	00010	62	63	74	85			00:00:00			00:00:00	

图 3-144

3.3.3　SAP系统与其他系统间数据传输

IDoc 传输支持 C、.Net、Java 等各种技术，但需要相关的第三方软件，如 Java 在 SAP 的 IDoc 操作中需要 sapidoc.jar，且在连接操作时需要 sapidocjco.jar，这里不做具体介绍。

3.4 RFC应用

在系统间通信过程中，需区分发送系统和接收系统。RFC 调用请求从发送系统（调用系统）中传至接收系统（被调用系统，也称远程系统或目标系统），发送请求的系统在通信过程中又称为 RFC 客户端，通信另一方则称为 RFC 服务器。RFC 客户端发起远程函数调用以执行 RFC 服务器提供的功能。

其中，调用系统和被调用系统均可以是 SAP 系统和非 SAP 系统，此外还可以在 SAP 系统内部将特定应用服务器指定为目标系统。

根据通信方向和系统类型，共有以下 3 种 RFC 通信：

- 两个独立的 SAP 系统之间的通信。
- SAP 系统作为调用系统，与外部远程系统（非 SAP ABAP 系统）通信。
- 外部系统作为调用系统，与 SAP 系统通信。

SAP 调用远程函数的能力是通过 RFC 接口系统（RFC interface system）实现的。根据调用方向的不同（SAP 系统调用其他模块或其他系统调用 SAP 模块），RFC 接口提供以下两种服务：

- ABAP 程序的调用接口。
- 非 SAP ABAP 程序的调用接口。

3.4.1 同步RFC方式的远程函数调用

同步 RFC 要求远程系统在调用时可用，调用程序的处理暂停，并等待远程函数调用的返回结果再继续。

对于表类型参数，只有扁平结构内表可以被传输，其他类型参数可以用于传输纵深类型以及 STRING 类型的数据，将内表通过 tables 参数进行传输将获取较快的传输速度，因为系统内部通过二进制格式进行传输而不是 XML 格式。

1．同步RFC调用示例

```
 DATA: gtd_bapi TYPE zscarr OCCURS 0.
 DATA: gth_bapi TYPE zscarr.
 DATA: datas TYPE zscarr-countryfr.
 DATA: mess TYPE c LENGTH 80.
*- 筛选数据
 SELECT a~countryfr a~cityfrom a~cityto a~fltime a~distance b~carrname a~distance a~distid
        INTO CORRESPONDING FIELDS OF TABLE gtd_bapi
 FROM spfli AS a INNER JOIN scarr AS b ON a~carrid = b~carrid .
*- 调用RFC
  CALL FUNCTION 'ZTEST_RFC'
   DESTINATION 'SAP_JAVA_TEST'
   TABLES
     p_table          = gtd_bapi
*---  这里异常不是函数自己定义的,是系统的
   EXCEPTIONS
     system_failure       = 1  MESSAGE mess
```

```
        communication_failure = 2  MESSAGE mess
        resource_failure          = 3.
*-  不成功则退出
    IF sy-subrc <> 0.
      WRITE: / mess.
      EXIT.
    ENDIF.
*-  打印调用后结果
    WRITE: / '----------------------'.
    LOOP AT gtd_bapi INTO gth_bapi.
      WRITE: / gth_bapi-countryfr.
  ENDLOOP.
```

如果远程函数处理时出现交互对话过程，将在当前调用程序的会话中先进行远程对话处理，处理结束后再继续调用程序，在远程系统为 SAP ABAP 系统的情况下，可以对功能模块进行正常的调试，即进入（step in）远程模块，系统在当前会话中进行远程调试，完成后再回到调用程序的调试界面。

2．同步调用的异常处理

同步调用 RFC 时，系统自动处理标准异常 communication_failure 和 system_failure。

若在系统中没有正确维护远程目标，或到远程系统的连接不能建立，系统将触发 communication_failure 异常。

若连接成功，但指定的远程函数不存在于目标系统中，系统将触发 system_failure。

若发生异常，不会引起中断（Short Dumpt），与该异常相关的中断的首行说明内容被传递到 message 附加项指定的变量 mess 中，变量类型为扁平的字符结构。

- 确保远程目标在 SM59 中被维护。
- 确保功能模块存在于该目标系统中。

3.4.2　异步RFC方式的远程函数调用

1．同步RFC调用示例

```
DATA: gtd_bapi TYPE zscarr OCCURS 0.
    DATA: gth_bapi TYPE zscarr.
    DATA: lwk_finished TYPE i VALUE 0.
    DATA: mess        TYPE c LENGTH 80.
    DATA: lwk_task  TYPE c LENGTH 8 VALUE 'Tttttt'.
*-  筛选数据
    SELECT a~countryfr a~cityfrom a~cityto a~fltime a~distance b~carrname a~distance a~distid
          INTO CORRESPONDING FIELDS OF TABLE gtd_bapi
    FROM spfli AS a INNER JOIN scarr AS b ON a~carrid = b~carrid.
*-  调用RFC
*--  The asynchronous RFC does not support communication with external systems or programs in other
programming languages.
    CALL FUNCTION 'ZTEST_RFC'
      STARTING NEW TASK 'Tt'
```

```
    DESTINATION 'NONE'
    PERFORMING back ON END OF TASK
    TABLES
      p_table             ·          = gtd_bapi
    EXCEPTIONS
      system_failure             = 1  MESSAGE mess
      communication_failure  = 2  MESSAGE mess.
*-  不成功则退出
  IF sy-subrc <> 0.
    WRITE: / mess.
    EXIT.
  ENDIF.
*-  打印调用之前的数据
  WRITE: / 'before----------------------'.
  LOOP AT gtd_bapi INTO gth_bapi.
    WRITE: / gth_bapi-countryfr.
  ENDLOOP.
*-  等待直到调用处理完毕
  WAIT UNTIL lwk_finished = 1 UP TO 10 SECONDS.
*-  打印调用处理后的数据
  WRITE: / 'after----------------------'.
  LOOP AT gtd_bapi INTO gth_bapi.
    WRITE: / gth_bapi-countryfr.
  ENDLOOP.


*&---------------------------------------------------------------------*
*&      Form  fm_callback
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->T        任务名称
*----------------------------------------------------------------------*
FORM back USING t.
  RECEIVE RESULTS FROM FUNCTION 'ZTEST_RFC'  KEEPING TASK
    TABLES
      p_table = gtd_bapi
    EXCEPTIONS
      system_failure             = 1 message mess
      communication_failure  = 2 message mess.
  .
  lwk_finished = 1.
ENDFORM.                "fm_callback
```

STARTING NEW TASK 指明异步调用模式，函数模块在新的工作过程中执行，也可以在后台模式处理异步 RFC 调用过程，在后台处理时，每个异步 RFC 调用也将占据一个对话工

作过程。

异步 RFC 调用过程只能使用 TABLES、EXPORTING（对应功能模块设定中的 Import 参数）和 EXCEPTIONS 参数。调用程序并不等待该模块的返回结果，因而不能指定 IMPORTING 参数，即不能直接接收从被调用功能中返回的信息。调用的同时只能接收系统标准异常，对于功能模块中抛出的其他特点异常则不能直接接收。

2．异步远程对话处理和调试

异步 RFC 调用过程也支持远程对话处理，在异步 RFC 调用时，系统总是打开新的工作进程，因而如果被调用功能模块包含对话程序（屏幕调用处理），则在该功能模块执行时，调用系统中将出现新的外部会话窗口，若调用功能时打开的会话数达到 6 个，则试图打开新窗口时触发 system_failure。

异步 RFC 调用时，对远程函数进行 step in 的调试过程，调用程序本身的调试会话不会进入远程调用函数内部，系统也打开新的会话，单独增加一个窗口，进行目标系统中远程调用函数的调试，将断点放在远程调用函数的 CALL FUNCTION 语句处，在异步 RFC 模式下进行调试。

3．异步RFC调用时接收结果

通过语句中的附加项 PERFORMING return_form on end of task 实现异步调用结果搜集，子程序必须存在于调用程序中，如果回调子程序中包含任何使当前程序执行中断的语句，call screen、submit、commit work、wait、RFC 调用及 W 或 I 类型的消息等，则不能成功返回结果。

通过语句子程序中 Receive results from function 从远程调用函数中接收结果。主程序运行期间，远程调用函数在其他工作过程中运行，因此系统变量 sy-subrc 和程序变量不会被更新，主程序将文本发送至输出列表的缓冲区中。

通过语句 WAIT UNTIL log_exp [UP TO sec SECONDS]用于异步 RFC 调用中等待结果的返回，该项必须与 performing 附加项配合使用，否则没有意义。

当满足 log_exp 条件后，程序继续执行，否则程序将挂起，并等待异步 RFC 调用的返回结果。当功能模块调用结束时，系统将自动执行回调子程序，在其中接收返回结果并设定相关的逻辑条件变量值，子程序结束后将回到 wait until 语句，在多个异步 RFC 调用存在的情况下，等待过程将反复重复，直到等待条件被满足，或不再有其他开发的异步 RFC 调用。

4．保持远程上下文

在 receive 语句中，通过 keeping task 附加项可以使已经加载的远程上下文保持至调用程序结束，直到远程连接终止为止。

远程调用函数中的交互对话处理都将保持至调用程序结束。通过 performing 接收结果的异步 RFC 调用不能进行内存滚动区的切换或进入内部模式，因为将导致异步返回结果无法传递给调用程序，可以通过 submit 或 call transaction 语句进行内存滚动区的切换。

3.4.3 并行RFC方式的远程函数调用

并行 RFC 方式的远程函数调用实际是异步 RFC 调用的应用之一。异步 RFC 调用实现并行处理：

- 异步 RFC 调用适用于多个 SAP ABAP 系统间的并行处理（不支持 SAP 系统和其他系统间的并行过程）。

- 在同一 SAP 系统内部使用异步 RFC 调用，将部分处理负载转移到其他的应用服务器方法时，将 RFC 目标指定为其他应用服务器。
- 若不显示指定异步 RFC 调用目标，在同一应用服务器内，可以通过本地异步 RFC 调用实现多个工作过程的并行处理。

注：事务 SM59 可将 SAP 系统中的应用服务器分配成不同的 RFC 分组，异步调用时，通过 DESTINATION 子句中的 IN GROUP 附加项指定一个已定义的 RFC 分组。若不显示指定 RFC 分组，还可以使用关键字 DEFAULT，系统从全部可用的应用服务器中选择一个处理。该项只能在当前 SAP 系统作为 RFC 目标时使用。

1．并行处理的前提条件

逻辑独立的工作单元，并行处理并不适合需要顺序进行的数据处理。

ABAP 要求函数模块必须设定为远程支持，被调用的函数模块不能包含使目标 BACK 的远程功能调用，调用程序不能在异步调用之后生成新的内部会话（即不能在 call function starting new task 语句后使用 submit 或 call transaction 语句），不能通过 call function starting new task destination in group 语句启动外部程序。

系统中的服务器至少有 3 个对话工作进程，否则触发 RESOURCE_FAILURE 异常

2．并行RFC调用示例

```
TYPES: BEGIN OF TASK_TYPE,
                  NAME TYPE STRING,
                  DEST TYPE STRING,
               END OF TASK_TYPE.

DATA: SND_JOBS  TYPE I,
      RCV_JOBS  TYPE I,
      EXC_FLAG  TYPE I,
      INFO      TYPE RFCSI,
      MESS      TYPE C LENGTH 80,
      INDX      TYPE C LENGTH 4,
      NAME      TYPE C LENGTH 8,
      TASK_LIST TYPE STANDARD TABLE OF TASK_TYPE,
      TASK_WA   TYPE TASK_TYPE.
DO 10 TIMES.
  INDX = SY-INDEX.
  CONCATENATE 'Task' INDX INTO NAME.
  CALL FUNCTION 'RFC_SYSTEM_INFO'
    STARTING NEW TASK NAME
    DESTINATION IN GROUP DEFAULT
    PERFORMING RFC_INFO ON END OF TASK
    EXCEPTIONS
      SYSTEM_FAILURE        = 1  MESSAGE MESS
      COMMUNICATION_FAILURE = 2  MESSAGE MESS
      RESOURCE_FAILURE      = 3.
```

```
  CASE SY-SUBRC.
   WHEN 0.
    SND_JOBS = SND_JOBS + 1.
   WHEN 1 OR 2.
    MESSAGE MESS TYPE 'I'.
   WHEN 3.
    IF SND_JOBS >= 1 AND
      EXC_FLAG = 0.
      EXC_FLAG = 1.
     WAIT UNTIL RCV_JOBS >= SND_JOBS
        UP TO 5 SECONDS.
    ENDIF.

    IF SY-SUBRC = 0.
     EXC_FLAG = 0.
    ELSE.
     MESSAGE 'Resource failure' TYPE 'I'.
    ENDIF.
   WHEN OTHERS.

    MESSAGE 'Other error' TYPE 'I'.
  ENDCASE.
ENDDO.
WAIT UNTIL RCV_JOBS >= SND_JOBS.

LOOP AT TASK_LIST INTO TASK_WA.
  WRITE: / TASK_WA-NAME, TASK_WA-DEST.
ENDLOOP.

*&---------------------------------------------------------------------*
*&      Form  rfc_info
*&---------------------------------------------------------------------*
FORM RFC_INFO USING NAME.

  TASK_WA-NAME = NAME.
  RCV_JOBS = RCV_JOBS + 1.
  RECEIVE RESULTS FROM FUNCTION 'RFC_SYSTEM_INFO'
   IMPORTING
    RFCSI_EXPORT = INFO
   EXCEPTIONS
    SYSTEM_FAILURE          = 1 MESSAGE MESS
    COMMUNICATION_FAILURE = 2 MESSAGE MESS.
  IF SY-SUBRC = 0.
   TASK_WA-DEST = INFO-RFCDEST.
  ELSE.
   TASK_WA-DEST = MESS.
```

```
    ENDIF.

    APPEND TASK_WA TO TASK_LIST.

  ENDFORM.              "rfc_info
```

注：WAIT UNTILL LOG_EXP UP TO N SECONDS，程序中断时间最长是 N 秒。

3.4.4 事务RFC方式的远程函数调用

同步、异步 RFC 过程中，每一个 RFC 调用在远程系统中构成一个独立的 LUW，而事务性 RFC 调用可以将多个逻辑上相关的远程调用绑定至同一个 LUW 上。在该 LUW 内，按顺序调用，要么执行所有数据库操作，要么完全回滚来取消全部数据库更新操作，保证 RFC 调用过程的完整性和数据一致性。

事务性 RFC 调用处理保证在程序到达 COMMIT WORK 语句时执行所有计划的更新，并且事务性 RFC 调用仅运行一次，该调用具有安全、可靠的特点。

1．事务RFC调用语法

（1）7.0 以前的事务 RFC

语法：

```
CALL FUNCTION func IN BACKGROUND TASK
                     [DESTINATION dest]
                     [EXPORTING  p1 = a1 p2 = a2... ]
                     [TABLES        t1 = itab1  t2 = itab2 ...]
                     [AS SEPARATE UNIT]
```

（2）7.0 以后的事务 RFC

语法：

```
CALL FUNCTION func IN BACKGROUND UNIT oref
                     [EXPORTING  p1 = a1 p2 = a2... ]
                     [TABLES      t1 = itab1  t2 = itab2 ...]
```

其中 oref 必须引用一个接口 IF_BGRFC_UNIT 实现类的对象（异常类捕捉 CX_BGRFC_INVALID_UNIT），在相邻两个 COMMIT WORK 之间出现的所有相同目标的异步调用都属于同一个 LUW。事务性 RFC 不需要等待每个单独的更新过程完成，调用程序就可以继续进行，直到事务结束，若调用发送时，远程系统不可用，调用将设为后台作业运行，不能从调用模块接收返回结果，不允许调用者进行与远程系统的交互性对话及测试。

不能直接或通过 RECEIVE RESULTS FROM FUNCTION 语句接收远程函数的返回结果，函数的接口不应指定任何 EXPORT 类型参数，否则在 CALL FUNCTION 语句中的 IMPORTING 参数将导致编译错误，事务性 RFC 运行的功能模块中不适合进行回调（调用远程目标 BACK）。

2．事务RFC调用示例

```
  DATA: gtd_bapi TYPE zscarr OCCURS 0.
  DATA: gth_bapi TYPE zscarr.
  DATA: lwk_finished TYPE i VALUE 0.
  DATA: mess  TYPE c LENGTH 80.
*-  筛选数据
```

```
  SELECT a~countryfr a~cityfrom a~cityto a~fltime a~distance b~carrname a~distance a~distid
         INTO CORRESPONDING FIELDS OF TABLE gtd_bapi
 FROM spfli AS a INNER JOIN scarr AS b ON a~carrid = b~carrid.
*-  调用 RFC
 CALL FUNCTION ' ZTEST_RFC '
   IN BACKGROUND TASK
   DESTINATION 'SAP_JAVA_TEST'
   AS SEPARATE UNIT
   TABLES
     p_table               = gtd_bapi
   EXCEPTIONS
     system_failure        = 1  MESSAGE mess
     communication_failure = 2  MESSAGE mess
     resource_failure      = 3.
*-  不成功则退出
 IF sy-subrc <> 0.
   WRITE: / mess.
   EXIT.
 ENDIF.
*-  加一些区别的数据
 gth_bapi-countryfr = 'LJ'.
 APPEND gth_bapi TO gtd_bapi.
*-  调用 RFC
 CALL FUNCTION 'ZBAPI_TEMP'
   IN BACKGROUND TASK
   DESTINATION 'SAP_JAVA_TEST'
   AS SEPARATE UNIT
   TABLES
     bapi_table            = gtd_bapi
   EXCEPTIONS
     system_failure        = 1  MESSAGE mess
     communication_failure = 2  MESSAGE mess
     resource_failure      = 3.
*-  不成功则退出
 IF sy-subrc <> 0.
   WRITE: / mess.
   EXIT.
 ENDIF.
*-  这里一定要提交
 COMMIT WORK.
```

在调用示例中直到 COMMIT WORK 语句才开始与远程系统进行联系，函数 ZTEST_RFC 是一个适合进行事务性 RFC 调用的远程函数模块，只进行远程系统中的数据库更新，不需要返回任何数据。

3．事务ID

每一个事务性 RFC 调用均通过一个独特的事务 ID（Transactional ID，TID），TID 存在于

数据库表 ARFCSSTATE 和 ARFCSDATA 中，ARFCSSTATE 记录 LUW 执行状态，ARFCSDATA 包含事务性 RFC 调用的输入数据。

作业执行过程中，从事务性 RFC 表中读取相关数据，与相应的事务性 RFC 进行通信，远程 LUW 成功执行，则相应的条目在表中删除。因此，若 LUW 运行成功，则无法重新执行，因而保证了仅执行一次。

若 COMMIT WORK 同时触发了本地更新操作，则事务性 RFC 调用在本地更新成功完成之后才开始运行。

4．设定作业开始时间

如果希望在特定时间启动 LUW，可以通过函数模块 START_OF_BACKGROUPTASK 设置作业开始的时间，必须在 LUW 内部且在第一个 CALL…IN BACKGROUNDTASK 语句之后和 COMMIT WORK 语句之前调用该函数模块。

对于某个调用需要创建独立的事务，不参与其他目标的事务性 RFC 调用的任何一个 LUW，可以通过 AS SEPARATE UNTIAL 附加项实现，各个 LUW 处理都是独立进行的，执行次序无法被保证。

5．出错处理机制

发生错误时，事务性 RFC 调用将启动重试机制或者回滚。通过 COMMIT WORK 执行远程调用时，不能建立到目标系统的连接，将在 ARFCSSTATE 表中记录当前状态，系统通过报表 RSARFCSE 根据当前的 TID 重新计划后台作业，准备进行下一次调用。默认时间间隔为 15 分钟，尝试至 30 次为止。

在 RFC 配置时（事物代码 SM59），通过以下路径进行尝试次数和时间间隔的设定如图 3–145 所示。



图 3–145

设定界面如图 3–146 所示。



图 3–146

若经过最大尝试次数，仍然无法连接，系统将停止调用报表 RSARFCSE，并将 ARFCSDATA 并状态更新为 CPICERR，默认为 8 天，将表中的响应条目删除，事物 SM59 可

以手动启动该条目。

执行出错时，A 类型的消息或者 RAISE 语句抛出异常，则 ARFCSSTATE 表将记录出错状态，整个 RFC 中的操作将取消，SM58 查看该错误并修改错误后，通过 SM58 重新启动出错的远程调用函数。

在远程调用函数中，功能模块 RESTART_OF_BACKGROUNDTASK 启动重试功能。

6．检查事务性RFC调用状态

每个 LUW 都用唯一的 TID 进行标识，有以下两种方法检查 TID 状态：

- ABAP 程序：在 CALL… IN BACKGROUND TASK 之后和 COMMIT WORK 之前调用函数“ID_OF_BACKGROUNDTASK”，获得 TID 后，用 FM“STATUS_OF_BACKGROUNDTASK”确定事务性 RFC 的状态。
- 联机确认：通过 SM58 显示并维护事务性 RFC 的 LUW 状态。

3.4.5 队列RFC方式的远程函数调用

确保事务性 RFC 调用中的 LUW 都按照其创建次序执行，需要使用队列 RFC 创建 LUW 序列。队列 RFC 是事务性 RFC 的增强版本，用于 SAP-SAP 及 SAP-非 SAP 系统间的连接。多个函数发送至目标系统中的三种可能情况（实际的连接过程仍然通过事务性 RFC 来实现，可以增加入站、出站队列）包括普通事务性 RFC、含出站队列的队列 RFC 和含出入站队列的队列 RFC，其中出站队列在队列 RFC 中是必须的。

出站调度器用于控制其他逻辑目标系统中的 LUW 执行，入站调度器控制本地系统中的队列 RFC 执行（目标 NONE 或 IN BACKGROUND TASK 实现为指定目标 RFC），事务 SMQS 可以配置出战的队列 RFC 序列，事务 SMQR 可以配置入站的队列 RFC 序列，需要在普通事务性 RFC 调用之前使用函数模块 TRFC_SET_QUEUE_NAME 来指明后续的事务性 RFC 所要插入的出战队列，通过函数 TRFC_SET_QIN_PROPERTIES 来完成入站队列。

队列 RFC 调用示例如下：

```
 DATA: gtd_bapi TYPE zscarr OCCURS 0.
 DATA: gth_bapi TYPE zscarr.
 DATA: lwk_finished TYPE i VALUE 0.
 DATA: mess TYPE c LENGTH 80.
*-  筛选数据
 SELECT a~countryfr a~cityfrom a~cityto a~fltime a~distance b~carrname a~distance a~distid
        INTO CORRESPONDING FIELDS OF TABLE gtd_bapi
 FROM spfli AS a INNER JOIN scarr AS b ON a~carrid = b~carrid.
*-  设置队列
 CALL FUNCTION 'TRFC_SET_QUEUE_NAME'
   EXPORTING
     qname                          = 'Q1'
*     NOSEND                        = ' '
*     TRFC_IF_SYSFAIL               = ' '
*     CALL_EVENT                    = ' '
*    EXCEPTIONS
*     INVALID_QUEUE_NAME            = 1
```

```
*      OTHERS                              = 2
          .
*-   调用 RFC
 CALL FUNCTION 'ZTEST_RFC'
    IN BACKGROUND TASK
    DESTINATION 'SAP_JAVA_TEST'
    TABLES
      p_table                    = gtd_bapi
    EXCEPTIONS
      system_failure             = 1  MESSAGE mess
      communication_failure      = 2  MESSAGE mess
      resource_failure           = 3.
*-   不成功则退出
 IF sy-subrc <> 0.
    WRITE: / mess.
    EXIT.
 ENDIF.
*-   提交
 COMMIT WORK.
```

（1）本地远程函数调用和回调

通过 DESTNATION 附加项的两个特殊目标 NONE 和 BACK，可以进行远程函数本地远程调用和回调。调用本地系统中的远程功能模块，即调用当前系统内部的远程函数，CALL FUNCTION 语句格式决定以远程调用或非远程调用方式运行。

本地调用时 call function 语句没有 destination、starting new task、in background task/unit 出现，不在单独的内存滚动区中运行，若当调用没有指定某些必选 exporting 参数时，会引发系统异常终止。

（2）远程回调

- 在服务器执行远程功能时，该服务器可以调用客户中随调用程序载入内存的功能模块，称为回调。
- 通过特殊目标 BACK 可以触发回调机制。

语法： CALL FUNCTION rfm_name
DESTINATION ‘BACK’

- 只有同步调用时，才可以在服务器中的被调用功能模块中使用 BACK 功能。

注：各种 RFC 调用总结如下:

- 同步 RFC 调用需要等待 RFM 返回结果；异步 RFC 不需要，但需要通过 RECEVICE 语句在回调子程序中接收结果。事务性 RFM 不能接收 RFM 返回值。
- 同步/异步 RFC 调用，服务器系统必须在调用时可用；事务性 RFC 无此限制。
- 同步/异步 RFC 调用过程，允许用户与远程系统进行交互对话；事务性 RFC 不允许。
- 同步/异步 RFC 调用过程，参数值将直接传输至支持远程调用的功能模块；事务性 RFC 的参数值暂时存储在数据库中。

■ 对于异步 RFC，只适用于 SAP 系统内部或之间的调用，不能通过 T 类型目标连接至外部系统。
■ 事务性 RFC、队列 RFC 支持状态查询。

3.4.6　外部系统RFC方式的远程调用

在与其他语言的程序进行通信时，SAP 将为各种语言及相关的操作系统提供 RFC 库及 SAP 连接器，以实现 RFC 接口，如：

■ 为 C 和 C++提供的 RFC SDK。
■ 为 Java 提供的 Java 连接器 JCo。
■ 为.NET 提供的.NET 连接器 NCo。
■ SAP 调用远程功能的能力是通过 RFC 接口系统（RFC interface system）实现的。

下面的示例通过 RFC 传递参数并调用 Java 端 Web Service 返回数据。

做成远程调用函数（Remote Function Module）的步骤如下：

步骤一：做成函数接口使用的结构体。

银行数据传入用结构体如图 3-147 所示。

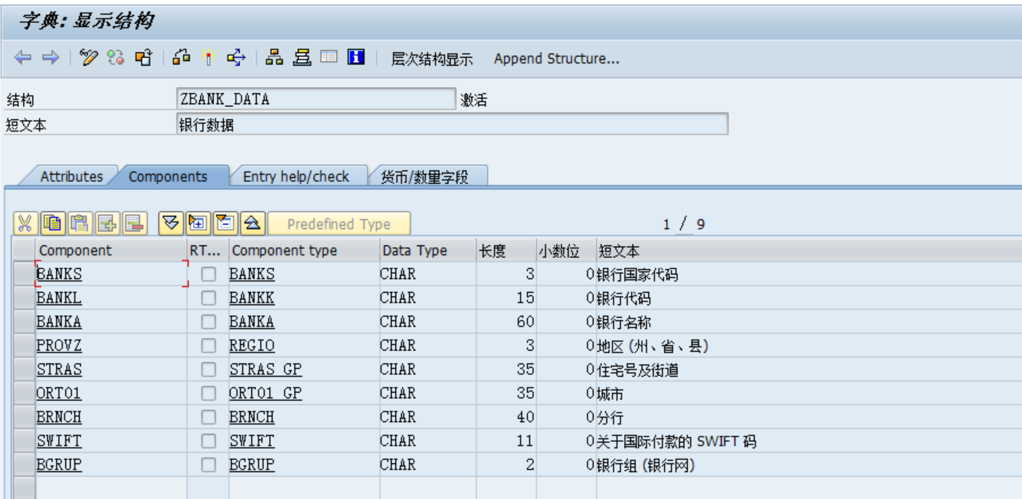


图 3-147

文件名传出用结构体如图 3-148 所示。

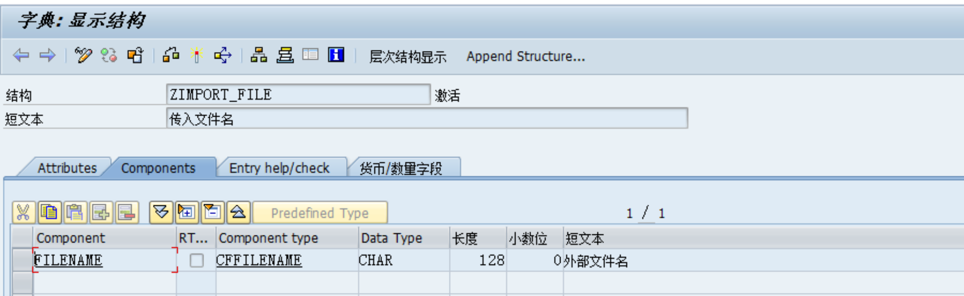


图 3-148

步骤二：做成函数。

函数组（Function Group）如图 3-149 所示。

图 3-149

远程函数属性（Attributes）标签页配置如图 3-150 所示。

图 3-150

远程函数传入（Import）标签页配置如图 3-151 所示。

图 3-151

远程函数传出（Export）标签页配置如图 3–152 所示。

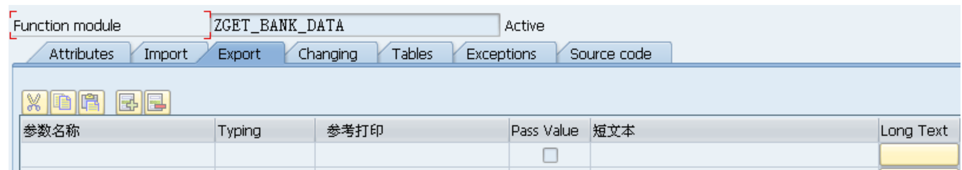


图 3–152

远程函数传入传出（Changing）标签页配置如图 3–153 所示。



图 3–153

远程函数传入传出（Tables）标签页配置如图 3–154 所示。

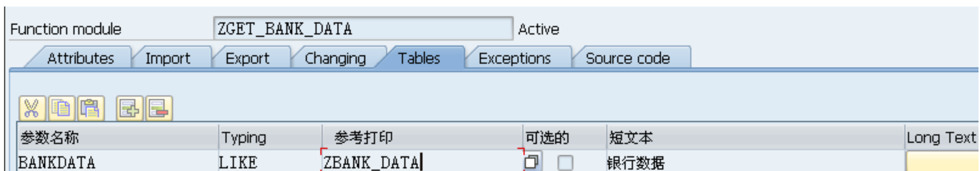


图 3–154

远程函数异常（Exceptions）标签页配置如图 3–155 所示。



图 3–155

远程函数原代码编辑如图 3–156 所示。

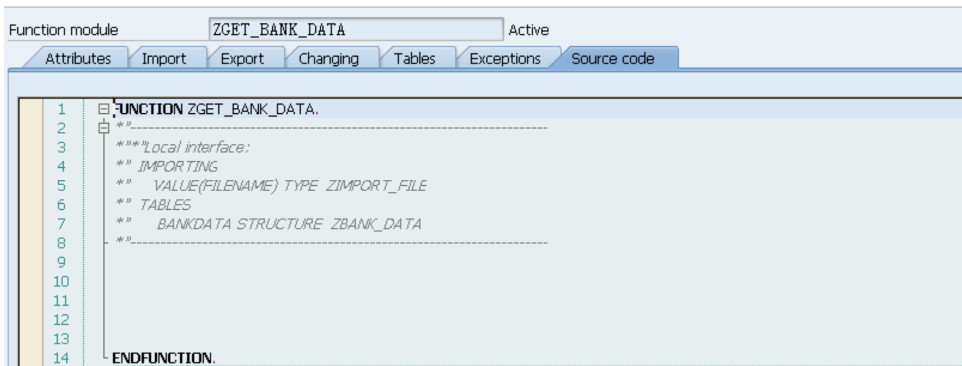


图 3–156

编辑测试程序源码如下：

```
*---------------------------------------------------------------------*
*
* Program ID                  ： ZRFC_JAVA
* Program Description         ： 通过 RFM 调用 JAVA
* Create  User                ： BCUSER
* Create  Date                ： 2011/02/20
*
*---------------------------------------------------------------------*
* Function Overview
* 通过逻辑文件选择某机器上的路径，读取上面的文件
* 并把结果返回给 R/3
*.
*---------------------------------------------------------------------*
* Change List          :
*     change NO.  Change Date   Change User   Change Detail
*     0000001     2011/02/20    BCUSER        创建程序
*---------------------------------------------------------------------*

REPORT  ZRFC_JAVA
   NO STANDARD PAGE HEADING
   LINE-SIZE 132
   LINE-COUNT 65(2)
   MESSAGE-ID ZTESTMSG.

*---------------------------------------------------------------------*
* CONSTANTS
*---------------------------------------------------------------------*
CONSTANTS: CNS_LOC_L TYPE C VALUE 'L'.

*---------------------------------------------------------------------*
* DATA
*---------------------------------------------------------------------*
DATA: STR_FILE TYPE ZIMPORT_FILE.
DATA  TBL_BANK TYPE TABLE OF ZBANK_DATA.
DATA  H_TBL_BANK TYPE  ZBANK_DATA.

*---------------------------------------------------------------------*
* PARAMETERS
*---------------------------------------------------------------------*
PARAMETERS:  P_TYPE  TYPE DXFILE-FILETYPE  OBLIGATORY.  "文件类型
PARAMETERS:  P_FILE  TYPE DXFILE-FILENAME  OBLIGATORY.  "文件

*---------------------------------------------------------------------*
* INITIALIZATION
*---------------------------------------------------------------------*
```

```
INITIALIZATION.

* 数据初始化
  PERFORM DATA_INIT.

*---------------------------------------------------------------------*
* AT SELECTION SCREEN OUTPUT
*---------------------------------------------------------------------*
AT SELECTION-SCREEN OUTPUT.

* 选择画面显示属性设定
  PERFORM SCREEN_ATTR_SET.

*---------------------------------------------------------------------*
* AT SELECTION SCREEN ON VALUE-REQUEST
*---------------------------------------------------------------------*
AT SELECTION-SCREEN ON VALUE-REQUEST FOR  P_FILE.

*  选择画面项目选择
  PERFORM FILE_INPUT_SELECT.

*---------------------------------------------------------------------*
*START-OF-SELECTION
*---------------------------------------------------------------------*
START-OF-SELECTION.

* 报表数据初始化
  PERFORM REPORT_DATA_INIT.
* 报表数据选择
  PERFORM REPORT_DATA_SELECT.
* 报表数据打印
  PERFORM REPORT_DATA_EXE.

*---------------------------------------------------------------------*
* TOP-OF-PAGE
*---------------------------------------------------------------------*
TOP-OF-PAGE.

* 报表页眉编辑
  PERFORM REPORT_HEADER_EDIT.

*&---------------------------------------------------------------------*
*&      Form  DATA_INIT
*&---------------------------------------------------------------------*
*       数据初始化
*---------------------------------------------------------------------*
```

```
FORM DATA_INIT .
  MOVE CNS_LOC_L TO  P_TYPE .
ENDFORM.                  " DATA_INIT
*&---------------------------------------------------------------------*
*&      Form  SCREEN_ATTR_SET
*&---------------------------------------------------------------------*
*       选择画面显示属性设定
*----------------------------------------------------------------------*
FORM SCREEN_ATTR_SET .
  LOOP AT SCREEN.
    IF  SCREEN-NAME = 'P_TYPE'.
      MOVE 0 TO SCREEN-INPUT.
      MODIFY SCREEN.
    ENDIF.
  ENDLOOP.
ENDFORM.                  " SCREEN_ATTR_SET
*&---------------------------------------------------------------------*
*&      Form  FILE_INPUT_SELECT
*&---------------------------------------------------------------------*
*       选择画面项目选择
*----------------------------------------------------------------------*
FORM FILE_INPUT_SELECT .
  CALL FUNCTION 'F4_DXFILENAME_4_DYNP'
    EXPORTING
      DYNPFIELD_FILETYPE      = 'P_TYPE'
      DYNPFIELD_FILENAME      = 'P_FILE'
      DYNAME                  = 'ZRFC_JAVA'
      DYNUMB                  = '1000'
      LOCATION                = 'A'
      SERVER                  = ''.
ENDFORM.                            " FILE_INPUT_SELECT
*&---------------------------------------------------------------------*
*&      Form  REPORT_DATA_INIT
*&---------------------------------------------------------------------*
*       报表数据初始化
*----------------------------------------------------------------------*
FORM REPORT_DATA_INIT .
 CLEAR: TBL_BANK[],H_TBL_BANK.
ENDFORM.                  " REPORT_DATA_INIT
*&---------------------------------------------------------------------*
*&      Form  REPORT_DATA_SELECT
*&---------------------------------------------------------------------*
*       报表数据选择
*----------------------------------------------------------------------*
FORM REPORT_DATA_SELECT .
  DATA L_FILE TYPE FILENAME-FILEINTERN.
```

```
  L_FILE = P_FILE.
  CALL FUNCTION 'FILE_GET_NAME'
   EXPORTING
*   CLIENT                  = SY-MANDT
    LOGICAL_FILENAME             = L_FILE         "伦理文件
*   OPERATING_SYSTEM             = SY-OPSYS
*   PARAMETER_1                  = ' '
*   PARAMETER_2                  = ' '
*   PARAMETER_3                  = ' '
*   USE_PRESENTATION_SERVER      = ' '
*   WITH_FILE_EXTENSION          = ' '
*   USE_BUFFER                   = ' '
*   ELEMINATE_BLANKS             = 'X'
  IMPORTING
*   EMERGENCY_FLAG               = EMERGENCY_FLAG
*   FILE_FORMAT                  = FILE_FORMAT
   FILE_NAME                     = STR_FILE-FILENAME "物理文件
  EXCEPTIONS
   FILE_NOT_FOUND                = 1
      .

  CALL FUNCTION 'ZGET_BANK_DATA'
   DESTINATION 'Z_TEST_DESTINATION'
   EXPORTING
    FILENAME = STR_FILE
   TABLES
    BANKDATA = TBL_BANK.

ENDFORM.                " REPORT_DATA_SELECT
*&---------------------------------------------------------------------*
*&     Form  REPORT_DATA_EXE
*&---------------------------------------------------------------------*
*      报表数据打印
*----------------------------------------------------------------------*
FORM REPORT_DATA_EXE .
  LOOP AT TBL_BANK INTO H_TBL_BANK.

   WRITE:  /2    H_TBL_BANK-BANKS,"国家
             7    H_TBL_BANK-BANKL,"银行代码
            18    H_TBL_BANK-BANKA,"银行名称
            38    H_TBL_BANK-PROVZ,"地区 (州、省、县)
            43    H_TBL_BANK-STRAS,"住宅号及街道
            63    H_TBL_BANK-ORT01,"城市
            73    H_TBL_BANK-BRNCH,"分行
            93    H_TBL_BANK-SWIFT,"关于国际付款的SWIFT 码
           100    H_TBL_BANK-BGRUP."银行组 (银行网)
```

```
  ENDLOOP.
ENDFORM.                " REPORT_DATA_EXE
*&---------------------------------------------------------------------*
*&      Form  REPORT_HEADER_EDIT
*&---------------------------------------------------------------------*
*       报表页眉编辑
*----------------------------------------------------------------------*
FORM REPORT_HEADER_EDIT .
 WRITE:  35  '银行文件数据一览',
         /2  '国家',
         7  '银行代码',
         18  '银行名称',
         38  '地区',
         43  '住宅号及街道',
         63  '城市',
         73  '分行',
         93  'SWIFT 码',
         100  '组'.

ENDFORM.                " REPORT_HEADER_EDIT
```

伦理文件路径设定如图 3-157 所示。



图 3-157

文件内容编辑如图 3-158 所示。

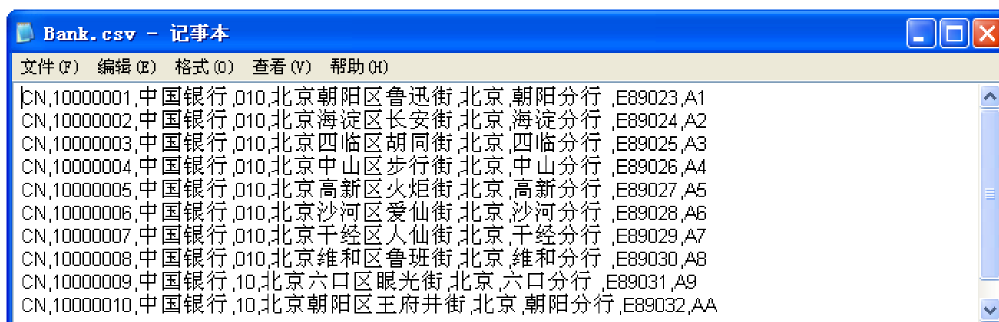


图 3-158

步骤三：配置 RFC。

RFC 技术设定标签页设定如图 3-159～图 3-161 所示。

RFC 目标系统 Z_TEST_DESTINATION
连接类型 T TCP/IP 连接 Description
说明
Description 1 RFC目标系统
Description 2 RFM调用JAVA用
Description 3

图 3-159

Administration | Technical Settings | 登录 | MDMP & Unicode | Special Options

Activation Type
Start on Application Server
Registered Server Program
Start on Explicit Host
Start on Front-End Work Station
Registered Server Program
Program ID Z_JAVA_TEST
Start Type of External Program
Default Gateway Value
Remote Execution
Remote Shell
Secure Shell

图 3-160

CPI-C Timeout
Default Gateway Value
Specify Timeout 60 Defined Value in Seconds
Gateway Options
Gateway Host 192.168.61.251 Delete
Gateway service SAPGW00

图 3-161

RFC 登录标签页设定如图 3-162 所示。

Administration | Technical Settings | 登录 | MDMP & Unicode | Special Options

Security Options
Logon Procedure
Send SAP Logon Ticket
Status of Secure Protocol
SNC Inactive Active
Authorization for Destination

图 3-162

RFC Unicode 标签页设定如图 3–163 所示。

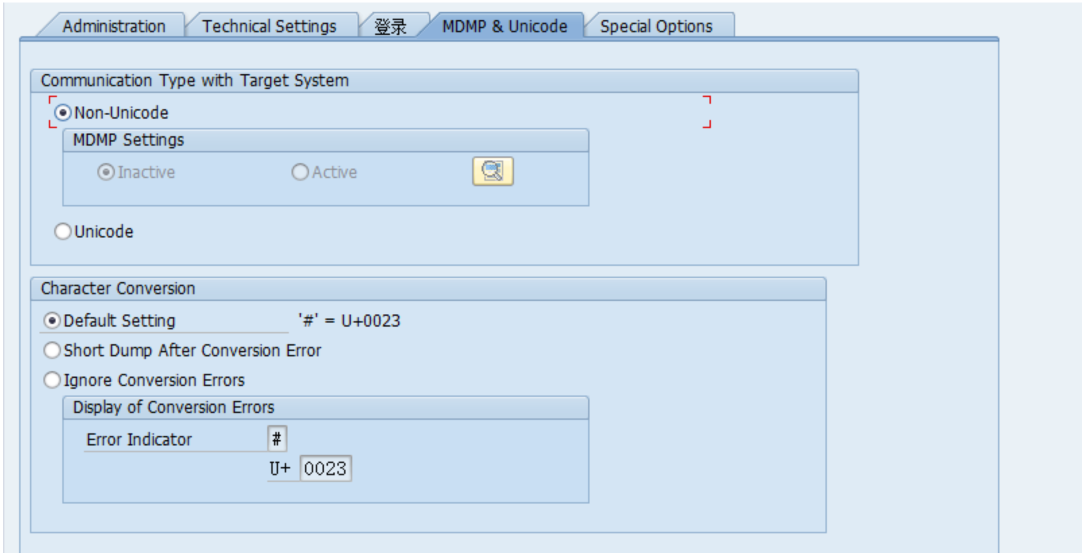


图 3–163

RFC 特殊属性标签页设定如图 3–164 和图 3–165 所示。

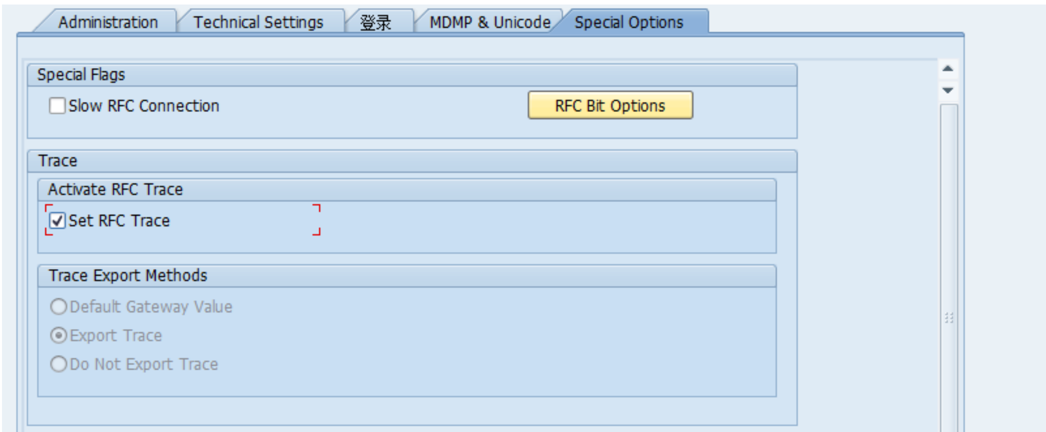


图 3–164

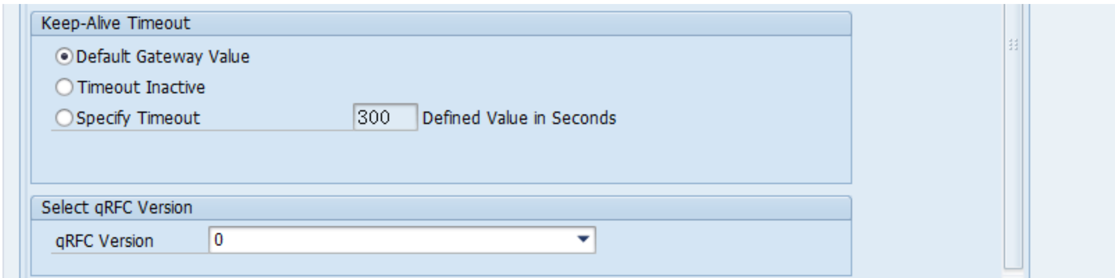


图 3–165

步骤四：编辑 JAVA 程序。

下载 SAP JAVA CONNECTORS 软件（sapjco.jar）及动态连接库 sapjcorfc.dll，安装 Java 开发工具 Eclipse，双击快捷图标或按照安装路径启动 Java 编辑工具，启动结果如图 3-166 所示

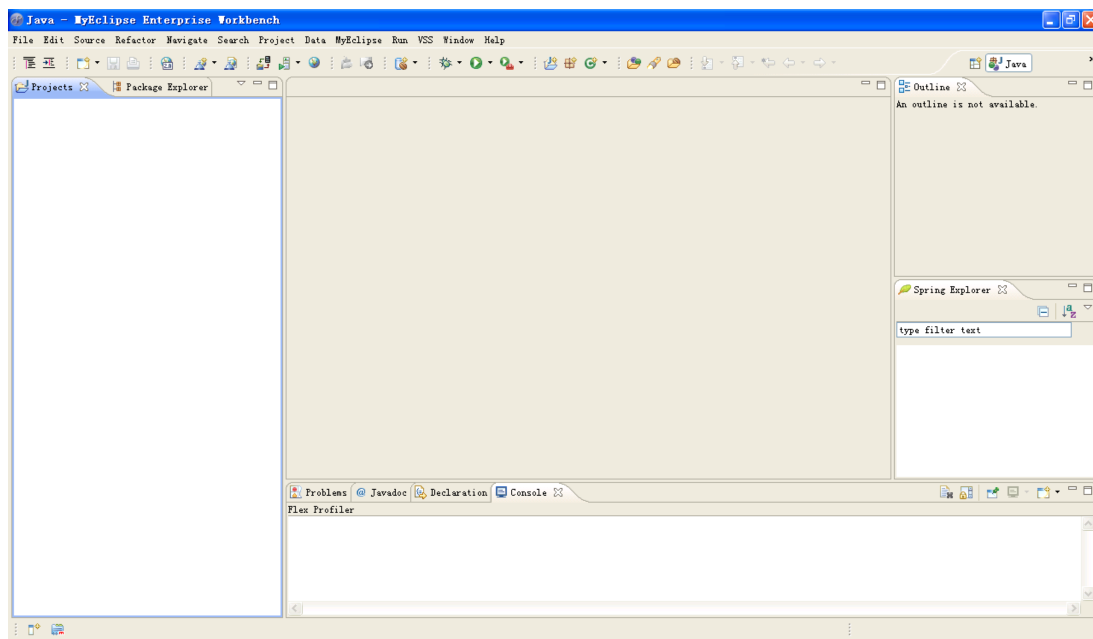


图 3-166

按照以下路径创建 Java 工程，如图 3-167 所示。

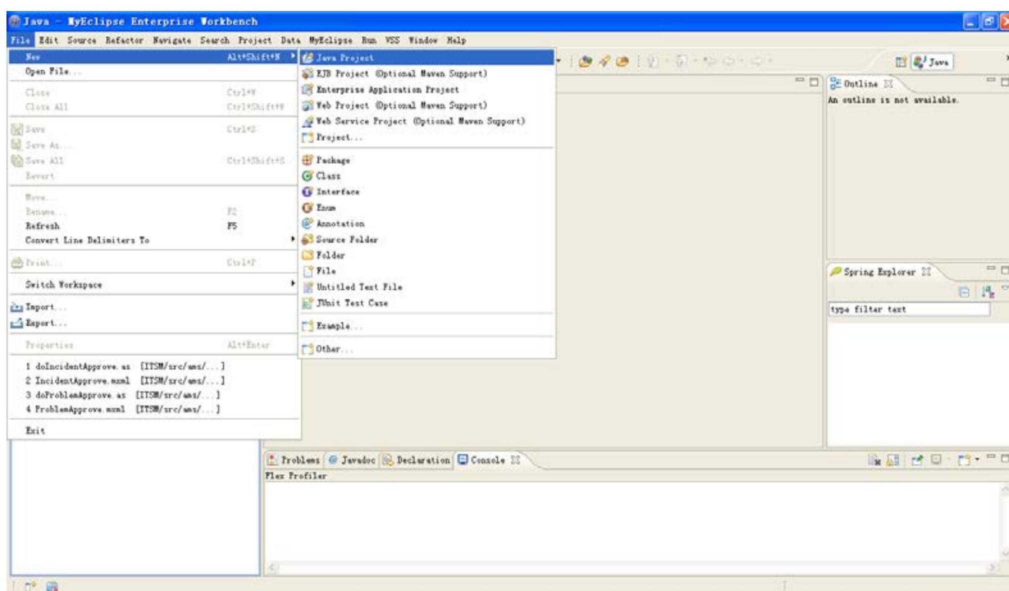


图 3-167

设定工程属性如图 3-168 所示。

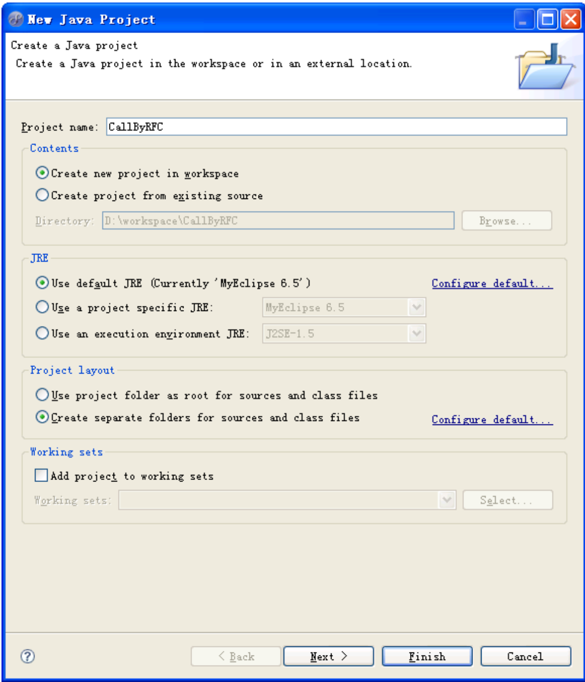


图 3-168

单击 Next > 按钮，编译环境设置如图 3-169 所示。

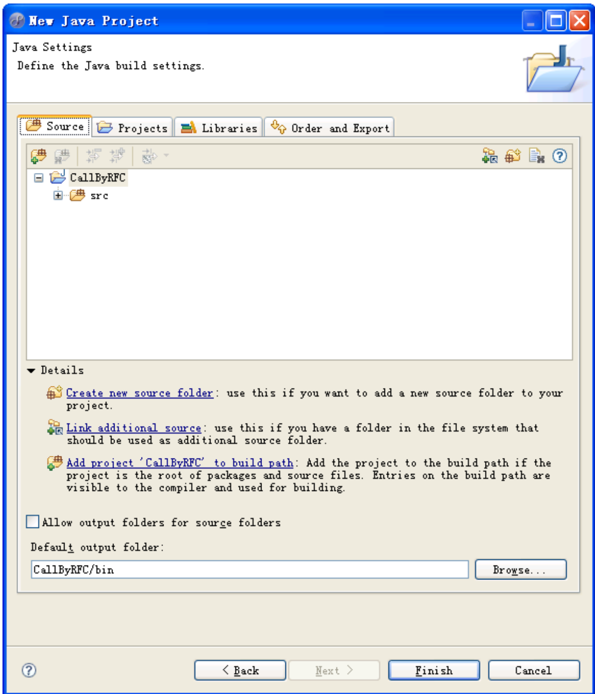


图 3-169

选择 Libraries 标签页，在配置库中添加 JCo 组件，如图 3–170 所示。

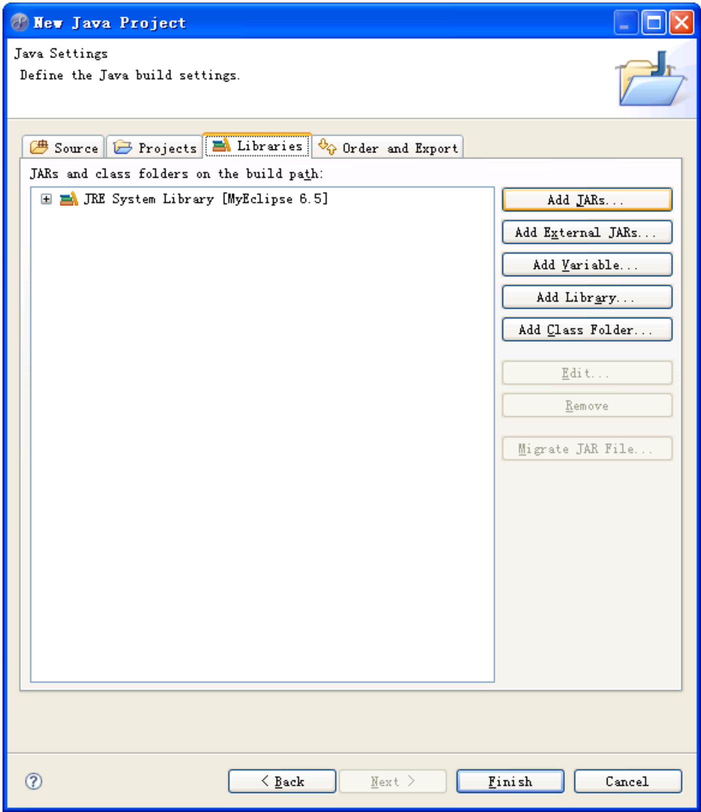


图 3–170

单击 Add External JARs... 按钮，如图 3–171 所示。

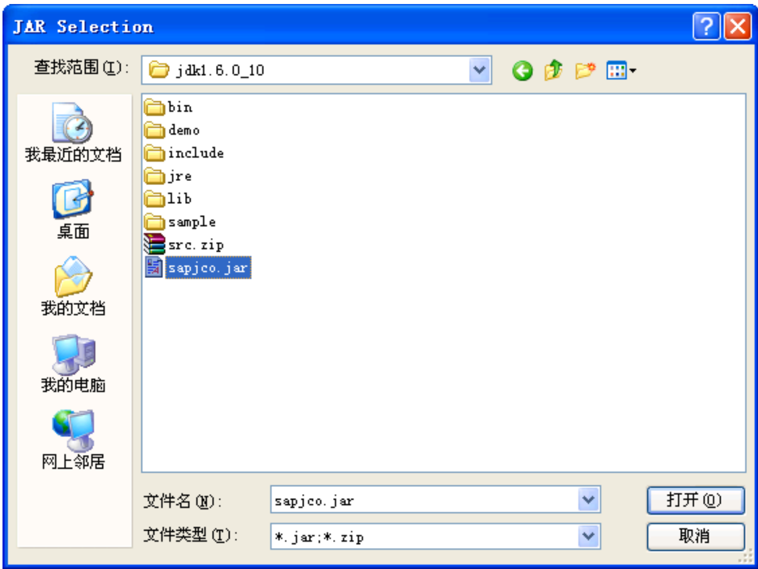


图 3–171

单击对话框中的打开(O)按钮，再单击Finish按钮，创建工程结果如图 3–172 所示。

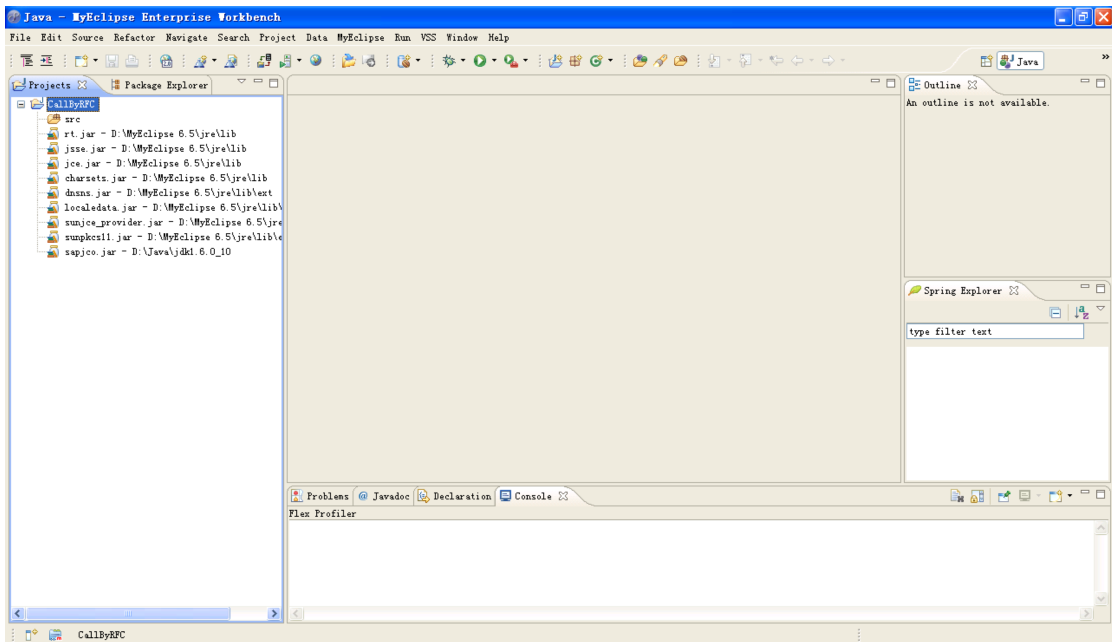


图 3–172

按照以下路径创建 Java 包，如图 3–173 所示。

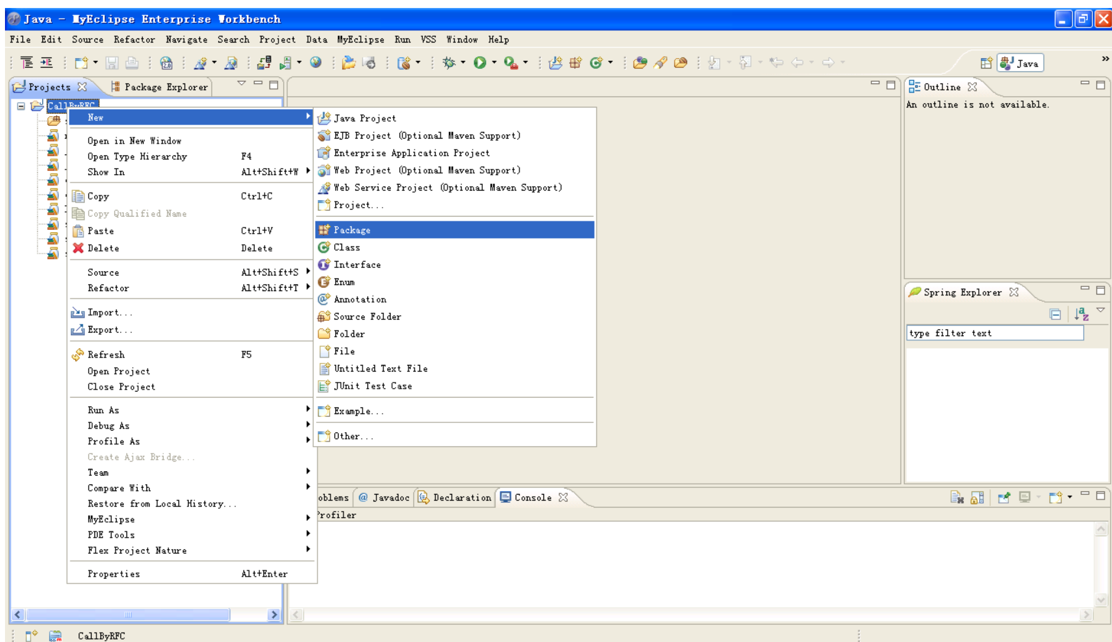


图 3–173

包名设定如图 3-174 所示。

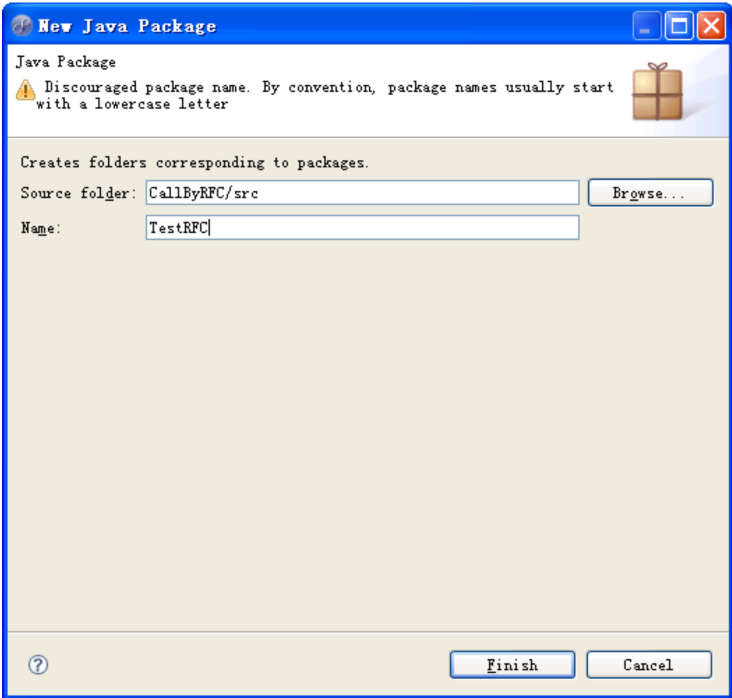


图 3–174

单击 Finish 按钮，按照以下路径为包添加 Java 类，如图 3–175 所示。

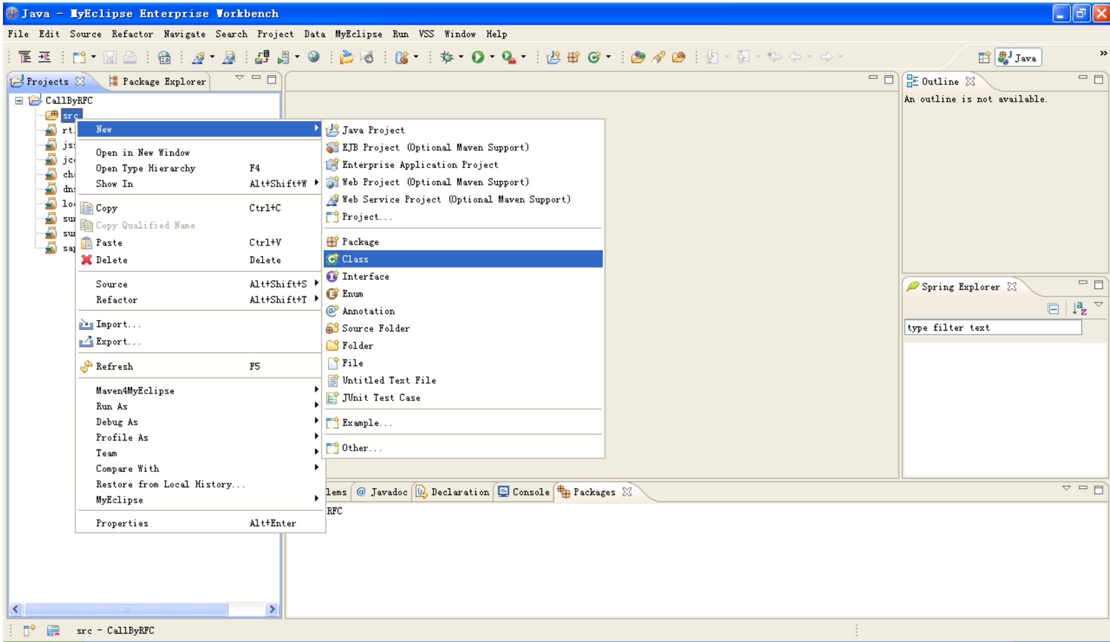


图 3–175

Java 类属性设定如图 3-176 所示。

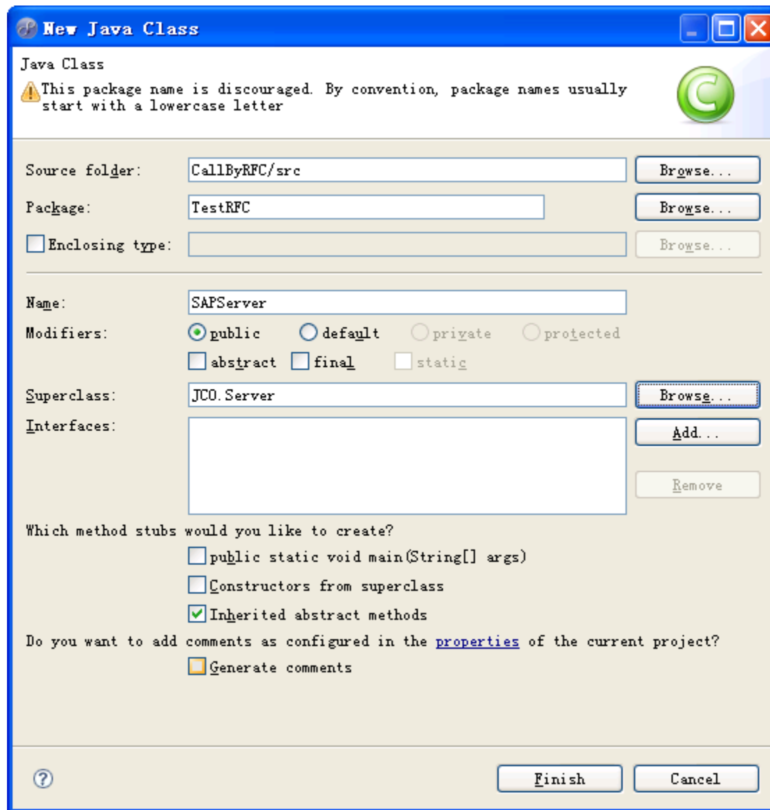


图 3-176

单击 Finish 按钮，编辑代码如下：

```
package TestRFC;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.util.regex.Pattern;
import com.sap.mw.jco.*;
import com.sap.mw.jco.JCO.Function;
public class SAPServer extends JCO.Server {
        public SAPServer（JCO.Repository repo）｛
                super（"192.168.61.251", "sapgw00", "Z_JAVA_TEST", repo）;
        }

        protected void handleRequest（Function arg0） throws Exception {
                //取得传入参数
                JCO.ParameterList importfile = arg0.getImportParameterList（）;
                //取得传入文件名
                String fileName = importfile.getStructure（"FILENAME"）.getString（"FILENAME"）.toString（）;
```

```
//编辑传入文件名
fileName = fileName.replaceAll（"\\\\", "\\\\\\\\"）;
File file = new File（fileName）;
JCO.Table banks = arg0.getTableParameterList（）.getTable（
        "BANKDATA"）;
//编辑传出表格
try {
    BufferedReader br = new BufferedReader（new java.io.FileReader（file））;

    String line = br.readLine（）;
    while （line != null） {
        Pattern comma = Pattern.compile（","）;
        String[] bank = comma.split（line）;
        banks.appendRow（）;
        banks.setValue（bank[0], "BANKS"）;   //银行国家代码
        banks.setValue（bank[1], "BANKL"）;   //银行代码
        banks.setValue（bank[2], "BANKA"）;   //银行名称
        banks.setValue（bank[3], "PROVZ"）;   //地区 （州、省、县）
        banks.setValue（bank[4], "STRAS"）;   //住宅号及街道
        banks.setValue（bank[5], "ORT01"）;   //城市
        banks.setValue（bank[6], "BRNCH"）;   //分行
        banks.setValue（bank[7], "SWIFT"）;   //关于国际付款的 SWIFT 码
        banks.setValue（bank[8], "BGRUP"）;   //银行组 （银行网）
        System.out.println（line）;
        line = br.readLine（）;
    }
    br.close（）;

} catch （IOException e） {

    e.printStackTrace（）;
    return;
}

//this.stop（）;

    }

}
```

按照以上步骤创建类 SAPLogon，编辑代码如下：

```
package TestRFC;
import com.sap.mw.jco.JCO;
public class SAPLogon {
        public JCO.Client mConnection;
        public JCO.Repository mRepository;
```

```
public SAPLogon（ String client,              //集团
                 String userid,              //用户
                 String password,            //密码
                 String language,            //语言
                 String ip,                  //IP
                 String system_number）      //系统编号
{
 try {
        mConnection = JCO.createClient（ client,              //集团
                                         userid,              //用户
                                         password,            //密码
                                         language,            //语言
                                         ip,                  //IP
                                         system_number）；    //系统编号

  mConnection.connect（）；
  mRepository = new JCO.Repository（"Lee", mConnection）；
  System.out.println（"SAP 连接成功"）；
   mRepository = new JCO.Repository（"my_repository", mConnection）；
   mConnection.disconnect（）；
 } catch （Exception ex） {
  ex.printStackTrace（）；
  System.exit（1）；
 }
}
}
```

同理，编辑类 StartServer，设定属性如图 3-177 所示。

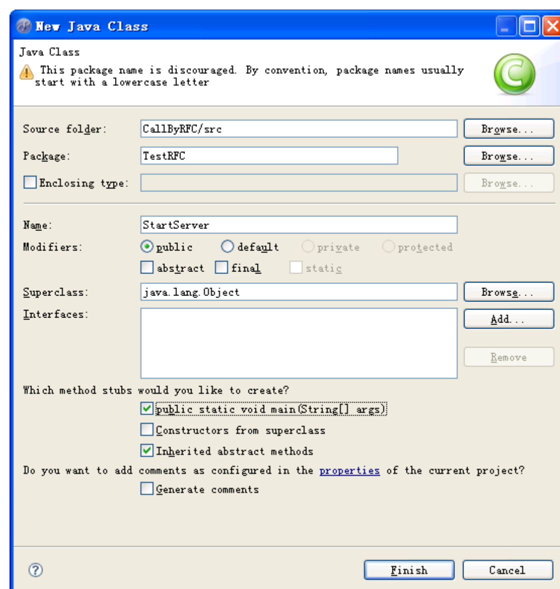


图 3-177

编辑代码如下：

```
package TestRFC;
public class StartServer {
        public static void main（String[] args）｛
                SAPLogon mySAP =
                  new SAPLogon（"000",        //集团
                                 "yidatec",    //用户
                                 "pylgkg",     //密码
                                 "1",          //语言
                                 "192.168.61.251",//IP
                                 "00"）；       //系统编号
                SAPServer myServer = new SAPServer（mySAP.mRepository）;
                myServer.start（）；
            }
}
```

注：将动态链接库 sapjcorfc.dll 拷贝到 D：\MyEclipse 6.5\jre\bin 或操作系统文件夹 C:\WINDOWS\system32 中。

步骤五：运行程序，调用 RFC，测试编辑结果。

在 Java 程序所在机器运行类 StartServer，如图 3−178 所示。

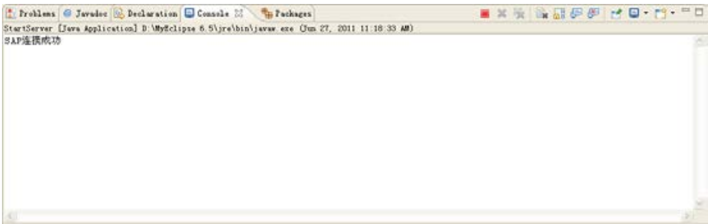


图 3−178

运行 ABAP 程序 ZRFC_JAVA，如图 3−179 所示。

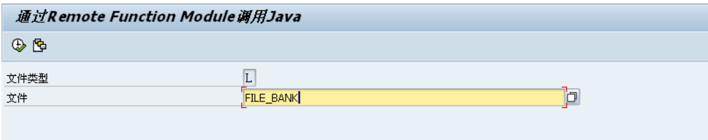


图 3−179

输入伦理文件名，单击按钮运行程序，结果如图 3−180 所示。

通过Remote Function Module调用Java

银行文件数据一览

国家	银行代码	银行名称	地区	住宅号及街道	城市	分行	SWIFT	组
CN	10000002	中国银行	10	北京海淀区长安街	北京	海淀分行	E89024	A2
CN	10000003	中国银行	10	北京四临区胡同街	北京	四临分行	E89025	A3
CN	10000004	中国银行	10	北京中山区步行街	北京	中山分行	E89026	A4
CN	10000005	中国银行	10	北京高新区火炬街	北京	高新分行	E89027	A5
CN	10000006	中国银行	10	北京沙河区爱仙街	北京	沙河分行	E89028	A6
CN	10000007	中国银行	10	北京千经区人仙街	北京	千经分行	E89029	A7
CN	10000008	中国银行	10	北京维和区鲁班街	北京	维和分行	E89030	A8
CN	10000009	中国银行	10	北京六口区眼光街	北京	六口分行	E89031	A9
CN	10000010	中国银行	10	北京朝阳区王府井街	北京	朝阳分行	E89032	AA

图 3−180

Web Service 后台 Log 输出如图 3-181 所示。

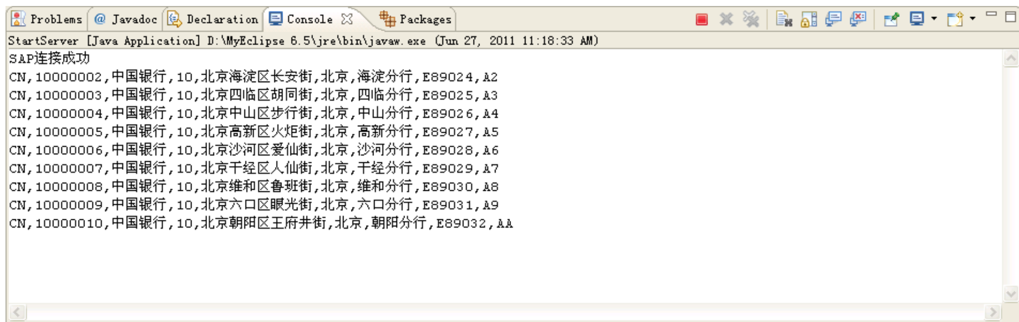


图 3-181

3.5 BAPI的应用

BAPI 业务应用编程接口（Business Application Programming Interfaces）业务应用编程接口，其作用可分为：

- 在 SAP 内部组件及非 SAP 组件之间的技术整合。
- 业务数据交换。
- SAP 通过该标准接口把整个系统链接为一个整体。
- BAPI 是 SAP 中心组件 R/3、ECC 和 BW 系统的入口之一。

BAPI 基于 RFC 技术实现，提供的整合扩展到了业务层面，外部程序可以通过 BAPI 访问 SAP 系统中的业务对象、数据、应用。

3.5.1 SAP业务对象

SAP 业务对象（Business Object）代表的是实际世界的对象，比如一个采购订单、一个物料数据等，包含业务数据，并描述了完整的业务过程。

SAP 的面向对象架构通过业务组件、业务对象类型、BAPI 等元素共同完成，每个 SAP 业务组件都提供相对独立的功能，其中包含业务对象类型和实例。

业务过程或业务场景可以在一个业务组件内部或者跨模块/系统实现。跨系统的业务过程也称为分布式业务过程，其集成通过 ALE 技术实现。业务对象类型是业务对象的定义和描述，它是面向对象架构的实现基础，也就是 SAP 系统中的“类”，它封装了业务功能和数据，一个业务对象即为一个业务实体，其中不仅包含业务对象本身的数据，还包含其功能的实现，根据业务规则来组织数据和业务过程。

业务对象的整体架构包含四层：

- 最内层：SAP 业务对象内核层，包含对象本身的内部数据结构及标准设定。
- 整合层：包含对象的业务逻辑。
- 接口层：提供该业务对象类型的外部访问接口，包括属性、方法、BAPI 方法、事件等。
- 外部访问层：支持对业务对象数据访问的具体技术、RFC/JCo/NCo 等。

业务对象仓库(Business Object Repository),是 SAP 业务对象类型和相关 BAPI 组织中心。业务对象仓库浏览器的事务代码是 SW03，业务对象在业务对象浏览器（SWO1）中定义，使用业务对象创建器可以浏览业务对象类型的组件。

接口中仅定义属性和方法的名称，但不能包含具体实现，SAP 业务对象不支持多重继承，即一个业务对象类型只能制定唯一的父类型，但一个对象可以实现多个接口。

BAPIs（Business Application Programming Interface）是为 SAP 业务对象提供的特殊方法，它们被存储在 SAP 系统的业务对象库中（Business Object Repository （BOR）），用于实施特殊业务任务。一个业务对象里可以有多个方法，每个方法就是一个 BAPI。

在 SAP 系统中，BAPIs 是具有 RFC 功能的模块，可以在函数编辑器（SE37）中做成。BAPIs 提供标准的业务接口，使得外部应用可以使用 SAP 的功能和数据。

利用 BAPIs 来访问 SAP 业务对象的客户端程序，可以是同一 SAP 系统的一部分，可以是外部系统的一部分，也可以是另一 SAP 系统的一部分。

要查看 SAP 系统中存在的 BAPIs，可以按如下路径寻找。登录 SAP 菜单→工具→业务框架→BAPI 浏览器，或调用事物代码：BAPI，如图 3-182 所示。

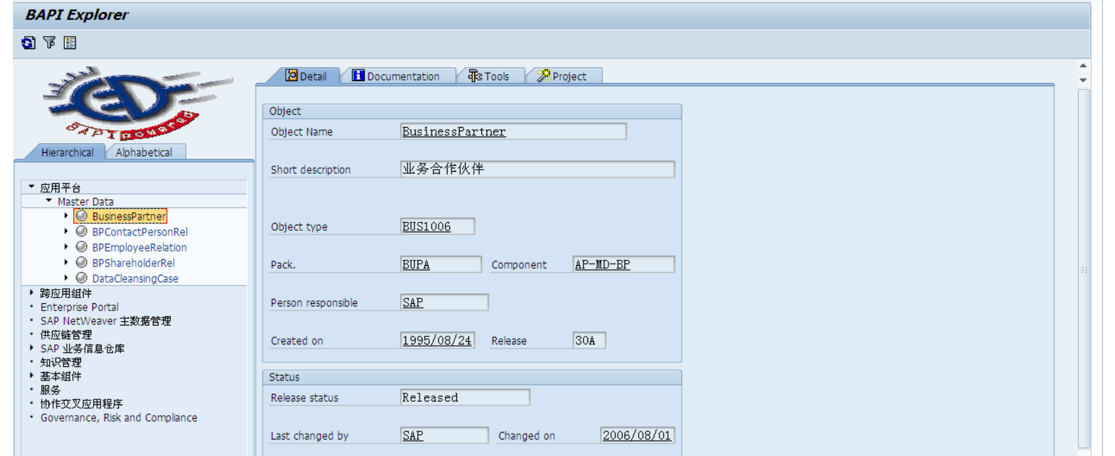


图 3-182

3.5.2 BAPI定义

1. 为BAPI定义构造体

BAPI 用输入参数定义如图 3-183 所示。

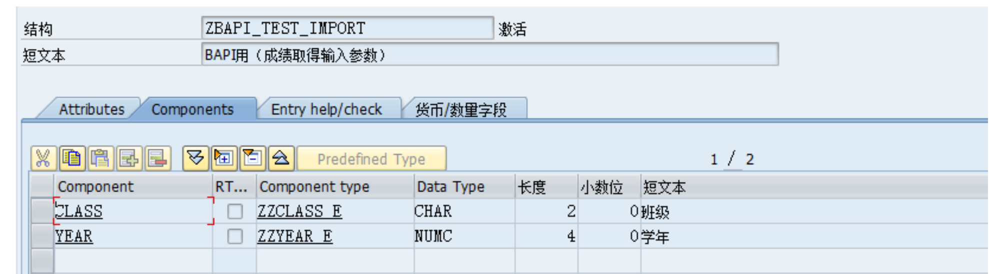


图 3-183

BAPI 用输出参数定义如图 3-184 所示。

结构 ZBAPI_TEST_EXPORT 激活

短文本 考试成绩信息

Attributes | Components | Entry help/check | 货币/数量字段

Predefined Type 1 / 8

Component	RT...	Component type	Data Type	长度	小数位	短文本
ZZCLASS		ZZCLASS_E	CHAR	2	0	班级
ZZYEAR		ZZYEAR_E	NUMC	4	0	学年
ZZSTDNO		ZZNUMBER_E	CHAR	5	0	学生番号
ZZNAME		ZZNAME_E	CHAR	30	0	姓名
ZZTENKOK		ZZTENKOK_E	DEC	3	0	语文成绩
ZZTENSAN		ZZTENSAN_E	DEC	3	0	数学成绩
ZZTENRIK		ZZTENRIK_E	DEC	3	0	自然成绩
ZZTENSHA		ZZTENSHA_E	DEC	3	0	思品成绩

图 3-184

- ■ 每个参数必须参照一个构造，必须以 BAPI 作为头字母，即使这个参数只有一个字段，它也必须参照一个构造的某个字段。
- ■ 一般情况下，项目中会对传入/传出的参数以结构形式定义。不推荐使用系统中已经存在的构造，因为当 BAPI 被释放后，BAPI 用的构造就会被冻结。

在 ABAP 数据字典（SE11）中建立构造体（具体步骤参见《SAP Dsbs 系列教程：ABAP 数据字典》）。

2．定义函数体（具体步骤参见[第 11 章：函数]）

（1）创建函数组

一个业务对象下的所有 BAPI 必须放在同一个函数组中，并且该函数组不能包含其他的函数，如图 3-185 所示。

显示功能组

函数组 ZTEST

函数组短文本 考试信息函数组

负责人员 BCUSER

开发类 ZABAP_TRAINING

应用程序 S

状态 已激活

Program status

Editor lock

Fixed point arithmetic

Unicode 检查激活

Translation of technical texts

变更需求(组织者) 主程序 功能组文档

图 3-185

（2）创建函数

属性标签页设定如图 3-186 所示。

Function module Z_BAPI_GET_TEST Active

Attributes | Import | Export | Changing | Tables | Exceptions | Source code

Classification

Function Group	ZTEST	考试信息函数组
Short Text	年级年度考试成绩	

Processing Type

- Normal Function Module
- Remote fähiger Baustein
- Update Module
  - 立即开始
  - Immediate Start, No Restart
  - 启动延迟
  - Coll.run

General Data

负责人员	BCUSER
Last Changed By	BCUSER
Changed on	08.06.2015
开发类	ZABAP_TRAINING
Program Name	SAPLZTEST
INCLUDE Name	LZTESTU01
Original Language	ZH
Not released	

Edit Lock

Global

图 3-186

注：在属性标签下选中 Last Changed By，否则这个函数模块不能通过 RFC 被调用，也不能作为一个 BAPI 使用。

输入（Import）标签页设定如图 3-187 所示。

Function module Z_BAPI_GET_TEST Active

Attributes | Import | Export | Changing | Tables | Exceptions | Source code

参数名称	Typing	参考打印	缺省值	可...	Pas...	短文本
IMPARAM	TYPE	ZBAPI_TEST_IMPORT		☐	☑	BAPI用（成绩取得输入参数）

图 3-187

输出（Export）标签页设定如图 3-188 所示。

Function Builder: Display Z_BAPI_GET_TEST

Pattern Pretty Printer Function Module Documentation

Function module Z_BAPI_GET_TEST Active

Attributes | Import | Export | Changing | Tables | Exceptions | Source code

参数名称	Typing	参考打印	Pass Value	短文本	Long Text
RETURN	TYPE	BAPIRETURN	☑	返回参数	

图 3-188

表（Tables）标签页设定如图 3-189 所示。

Function Builder: Display Z_BAPI_GET_TEST

Pattern Pretty Printer Function Module Documentation

Function module Z_BAPI_GET_TEST Active

Attributes | Import | Export | Changing | Tables | Exceptions | Source code

参数名称	Typing	参考打印	可选的	短文本	Long Text
TESTDATA	LIKE	ZZTEST_S	☐	考试成绩信息	

图 3-189

■ 每个 BAPI 必须有 RETURN 参数，这个参数可能是 Export Parameter，也可能是 Export Table。它的类型一般是 BAPIRET2 或 BAPARETURN。
■ BAPI 中不可以使用 Exceptions，错误消息都放到 RETURN 中。
■ 所有的 Currency 字段都参照 BAPICURR 域。
■ 所有的日期字段都是 YYYYMMDD 的格式。

源代码如下：

```
FUNCTION  Z_BAPI_GET_TEST.
*"----------------------------------------------------------------------
*"*"Local  interface：
*"   IMPORTING
*"       VALUE（CLASS）、 TYPE   ZZCLASS_E  DEFAULT  '1A'
*"       VALUE（YEAR）、 TYPE   ZZYEAR_E  DEFAULT  '2009'
*"   EXPORTING
*"       VALUE（RETURN）、 TYPE   BAPIRETURN
*"   TABLES
*"        TESTDATA  STRUCTURE   ZZTEST_S
*"----------------------------------------------------------------------
  SELECT              * FROM    ZZTEST_V
     INTO  CORRESPONDING FIELDS  OF  TABLE  TESTDATA
            WHERE    ZZCLASS    =  CLASS
            AND       ZZYEAR     =  YEAR.
  IF  SY-SUBRC <> 0.
     RETURN-TYPE   =  'S'.
     RETURN-CODE    =  '10005'.
     RETURN-MESSAGE  =  '没有相关数据！'.
  ENDIF.


ENDFUNCTION.
```

■ BAPI 不能执行 COMMIT WORK 及 ROLLBACK 命令（用函数 BAPI_TRANSACTION_COMMIT 和 BAPI_TRANSACTION_ROLLBACK 代替）。
■ BAPI 常用于更新数据库和取值。
■ BAPI 不可以显示画面，并且 BAPI 调用的函数也不能显示画面。

（3）保存、有效化并发布函数

发布路径如图 3-190 所示。

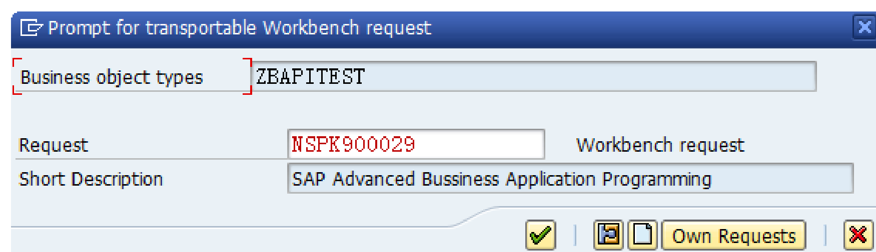


图 3-190

3．创建业务对象（事务代码：SWO1）

步骤一：业务对象制作程序的初始画面。

在“对象/接口类型”栏位中填入业务名称，单击 创建 按钮，如图 3-191 所示。



图 3-191

在相应的栏位填入说明和程序信息，如图 3-192 所示。



图 3-192

单击上图中的按钮，如图 3-193 所示。

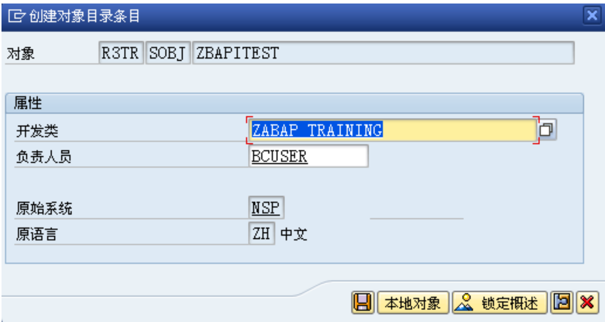


图 3-193

单击上图中的保存按钮，如图 3-194 所示。

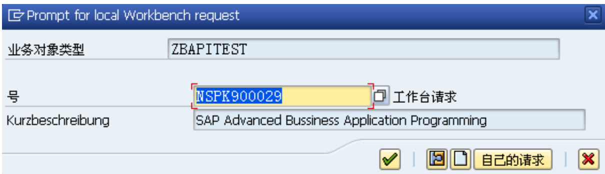


图 3-194

单击上图中的按钮。

步骤二：业务对象制作程序的对象编辑画面和添加 API 的方法。

按照以下路径选中相应菜单项，如图 3–195 所示。

在“Function Module”栏位中填入已定义函数，如图 3–196 所示。



图 3–195

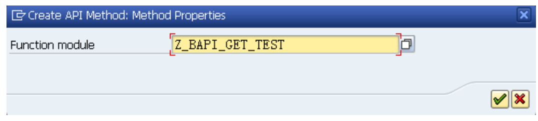


图 3–196

单击上图中的按钮，如图 3–197 所示。

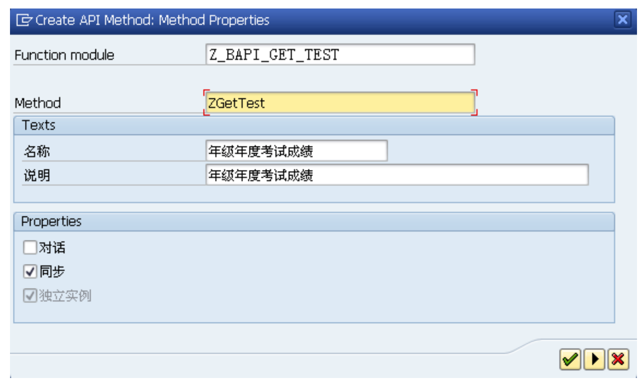


图 3–197

- [对话]复选框被选中表明该 BAPI 包含用户交互对话，否则不包含。
- [同步]复选框被选中表明该 BAPI 是同步调用。否则，是异步调用。
- [独立实例]复选框被选中表明该 BAPI 是实例非依存的。否则，是实例依存的。

注：比较低的版本有[返回结果]复选框，设定方法将返回一个参数作为结果，在结果类型中设定值列表，只有同步方法才能设定返回结果。

单击上图中的按钮，如图 3–198 所示。

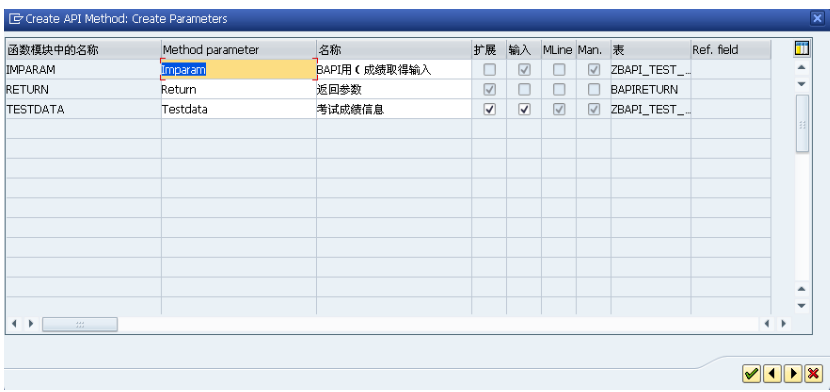


图 3–198

■ 「扩展」复选框被选中表明该参数是传出参数。
■ 「输入」复选框被选中表明该参数是传入参数。
■ 「Mline」复选框被选中表明该参数是一个表。
■ 「Man.」复选框被选中表明该参数是必选的。
■ 创建者必须指定表是传入参数还是传出参数，或者两者都是。

单击上图中的按钮，如图 3-199 所示。

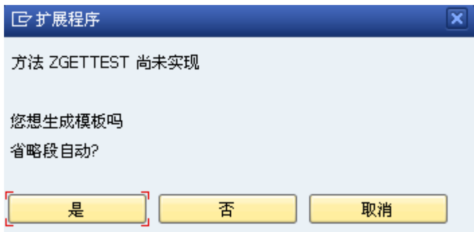


图 3-199

单击上图中的 是 按钮，如图 3-200 所示。



图 3-200

步骤三：保存、实现并生成业务对象。

1）单击标准工具栏中的按钮，如图 3-201 所示。



图 3-201

2）选中画面中的业务对象 ZBAPITEST，按照以下路径将其实装，如图 3-202 所示，其效果如图 3-203 所示。

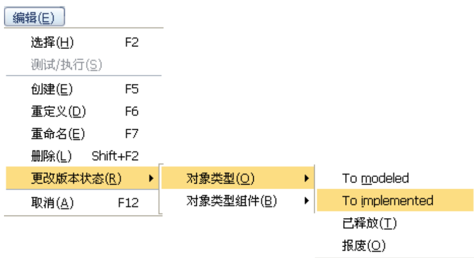


图 3-202



图 3-203

3）选中方法 ZBAPITEST_GET.ZGetTest，按照以下路径将其实装，如图 3-204 所示，其效果如图 3-205 所示。

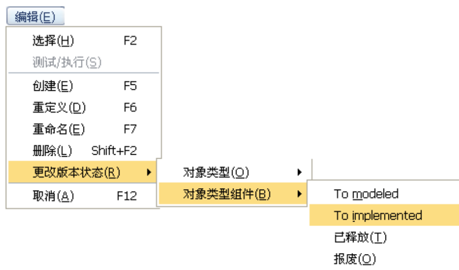


图 3-204



图 3-205

注：以上两个步骤缺一不可，开发人员需要注意。

4）单击应用工具栏中的按钮，生成业务对象，如图 3-206 所示。



图 3-206

步骤四：BAPI 确认。

运行事务代码 BAPI，确认 BAPI 生成结果，如图 3-207 所示。

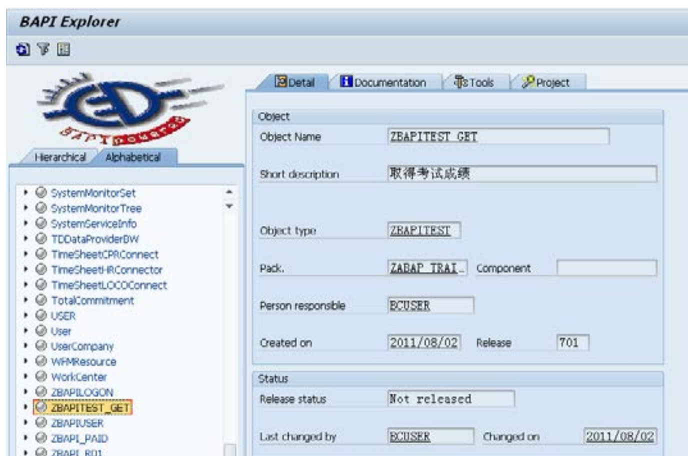


图 3-207

3.5.3　BAPI调用

BAPI 是在 BOR 中为 SAP 业务对象类型或接口类型定义的特殊方法，通过 RFC 支持的 ABAP 功能模块实现，和业务对象关联的远程函数模块也称为 BAPI 函数模块，俗称 BAPI。

BAPI 支持同步、异步的数据通信过程，通过基于 RFC 协议实现的 BAPI 接口，可以从应用层直接对 SAP 业务对象进行访问。

BAPI 不存在表现层，调用 BAPI 的第三方负责显示传入 BAPI 或 BAPI 返回数据，BAPI 建立在 RFC 协议基础上，外部语言需要进行 RFC 调用，即通过外部 RFC 接口来调用 BAPI。

通过 SAP 业务连接器，还支持从 Internet 直接访问 BAPI，可以从 BAPI 调用中创建 XML

文档，或者将入站 XML 文档转换为 BAPI 调用，实现 SAP 到 Internet 集成：

■ 集成 SAP 系统中已存在的业务应用及组件。
■ 集成已实施的 SAP R/3、ECC 和其他 SAP 系统。
■ 集成外部应用与 SAP 系统。
■ 网络应用集成（SAP 业务连接器和互联网应用组件）。
■ BAPI 作为系统业务数据导入、导出方式之一，应用于数据传输、ALE 数据分布过程。

1．BAPI查找

■ 从业务对象创建器查找 BAPI（事务代码 SWO1）。
■ 通过 BAPI 浏览器查找 BAPI（事务代码 BAPI）。
■ 利用 BAPI 对应的功能模块命名规则查找 BAPI_<bo>_<method>（事务代码 SE37）。

2．BAPI更新

BAPI 内部则通常不包含 COMMIT WORK 和 ROLLBACK WORK，在 BAPIs 里多数情况下不使用 COMMIT / ROLLBACK WORK 语句来修改数据库的数据，而是调用 BAPI 的 Object BapiService。在 BAPI 内部，数据库更新操作必须通过同步或异步更新过程实现，BAPI 内部也不触发新的 LUW，因此内部程序代码不包含 CALL TRANSACTION、SUBMIT REPORT 和 SUBMIE REPORT AND RETURN。

多个 BAPI 操作需要遵循以下原则：

■ 当有更新、修改、删除一个业务对象实例时，对实例进行另外读取操作的 BAPI 只能访问上一个 COMMIT WORK 执行后的最新数据。
■ 不允许在一个 LUW 中创建一个新实例，然后再修改它。
■ 可以创建多个相同对象类型的不同实例。

BAPI 事务中的数据库提交和回滚：

■ 必须通过调用 SAP 标准业务对象 BapiService 的 BAPI 方法 bapiservice.transactioncommit 和 bapiservice.transactionrollback 实现。
■ 利用远程调用函数：bapi_transaction_commit 和 bapi_transaction_rollback 实现。

在外部调用 bapiservice.transactioncommi 方法前，外部程序调用 BAPI 并不触发数据库提交，RFC 一旦建立，即外部程序登录 SAP 系统后，就一直保持连接。

3．BAPI调用特点

BAPI 调用完全从属于 RFC 的调用特性，支持 RFC 的本地调用及远程调用。

■ 支持 Remote Function Call（RFC）。
■ 它被分配为 Business Object Repository 中 SAP Business Object 的一个方法。
■ 没有 Exception 捕捉。
■ BAPI 调用是否成功是通过参数 RETURN 返回有关信息的。RETURN 用于 BAPI 返回错误和成功的消息，它可以是 BAPIRETURN、BAPIRETURN、BAPIRET1、BAPIRET2 和 BAPIRET2_FIX 中的任意一种类型。

4．BAPI的使用范围

■ 用 VB/JAVA/C++等语言编写非 SAPGUI 程序访问和处理 SAP 数据。
■ SAP 不同部件之间通讯。

■ 通过 SAP Business Connector 或 Internet Application Components 与 Internet 程序集成。

5．JCO实例做成

与 RFC 的 Java 调用一样，下载 SAP JAVA CONNECTORS 软件（sapjco.jar）及动态连接库 sapjcorfc.dll，其中 DLL 文件一定要放到系统路径下，XP 系统中一般是 C：\WINDOWS\system32。安装 Java 开发工具 Eclipse，双击快捷图标或按照安装路径启动 Java 编辑工具。

步骤一：启动 Java 编辑器，按照以下路径创建 Java 工程，如图 3-208 所示。

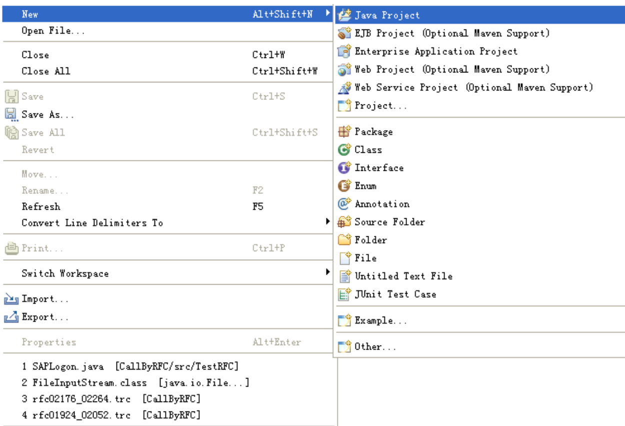


图 3-208

工程属性设定如图 3-209 所示。

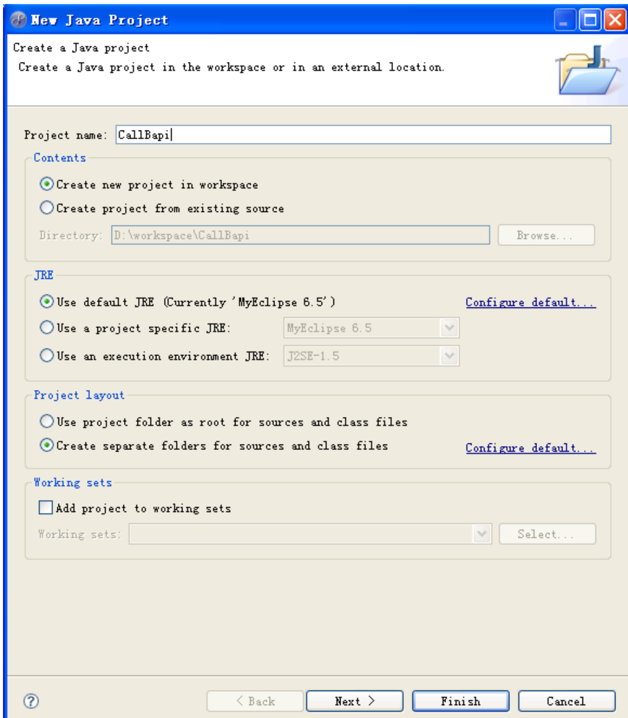


图 3-209

单击 Next > 按钮，编译环境设置如图 3-210 所示。

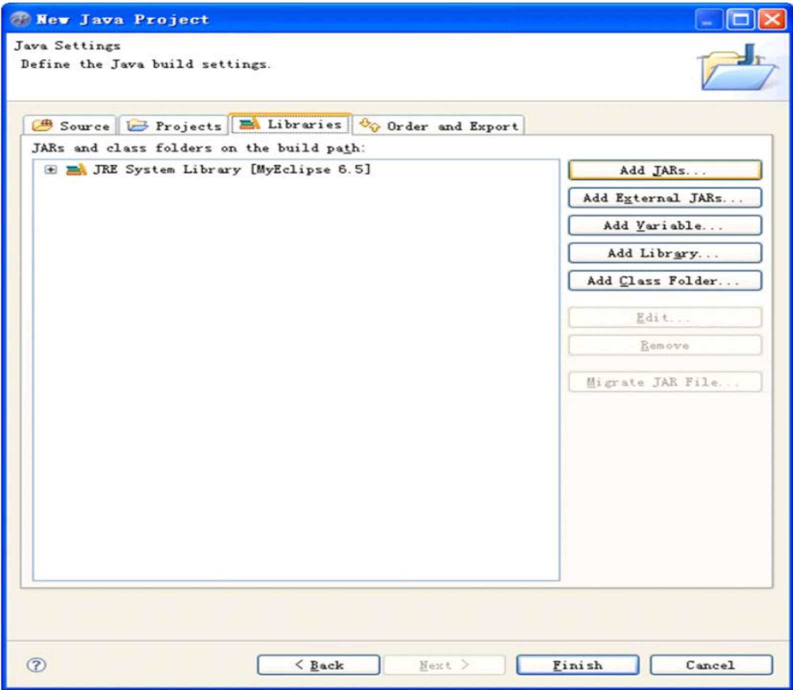


图 3-210

选择 Libraries 标签页，在配置库中添加 JCo 组件，单击 Add External JARs... 按钮，如图 3-211 所示。

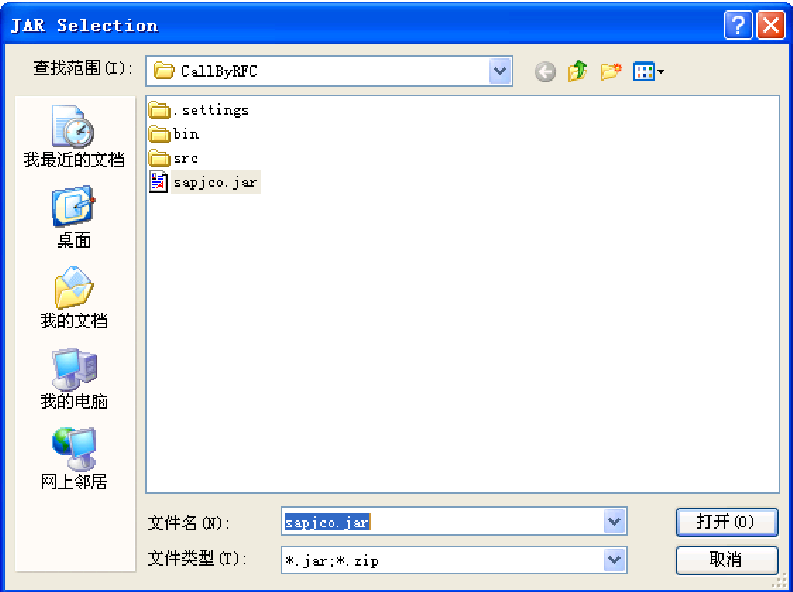


图 3-211

单击对话框中的打开(O)按钮，设置结果如图 3–212 所示。

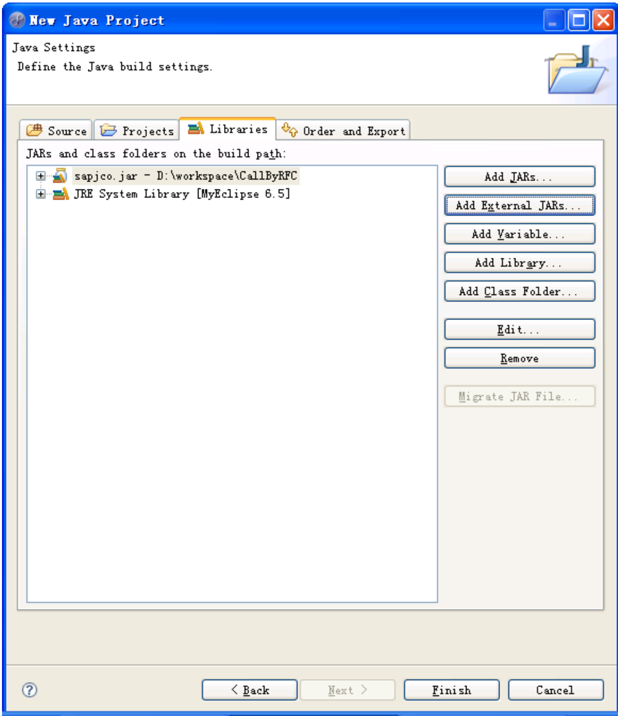


图 3–212

单击Finish按钮，创建工程后按照以下路径创建 Java 包，如图 3–213 所示。

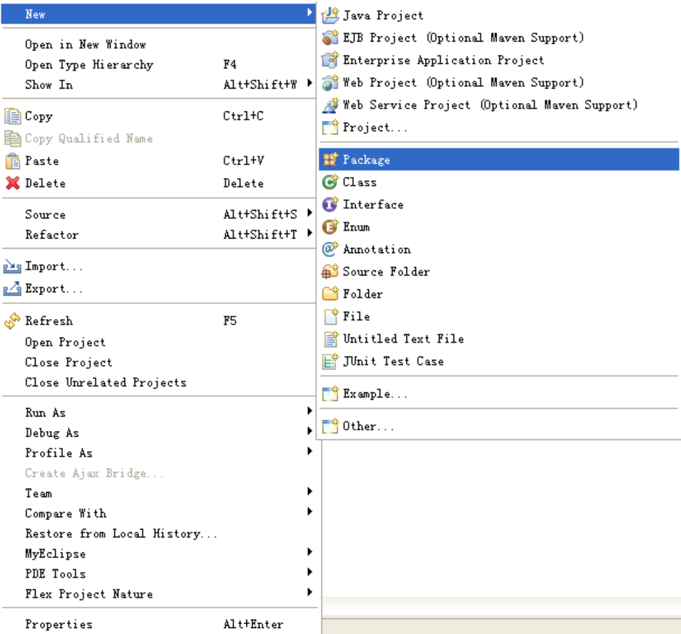


图 3–213

包名设定如图 3–214 所示。

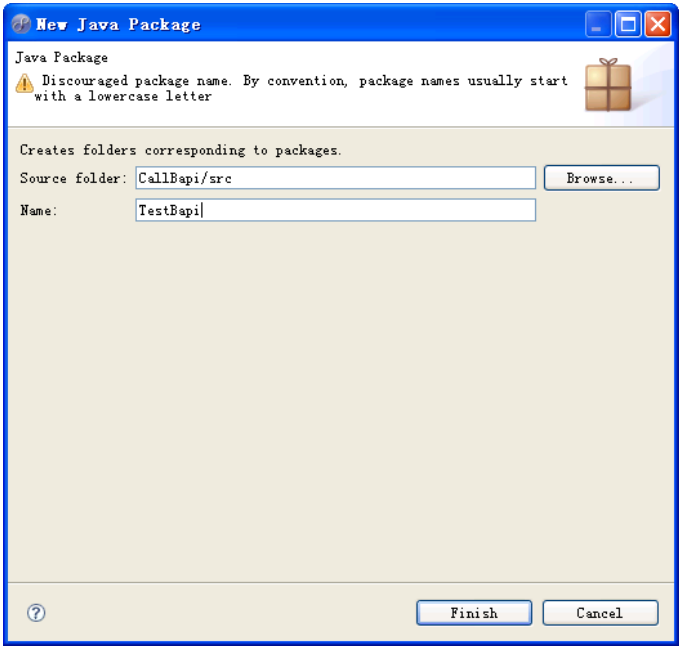


图 3–214

单击 Finish 按钮，按照以下路径为包添加 Java 类，如图 3–215 所示。

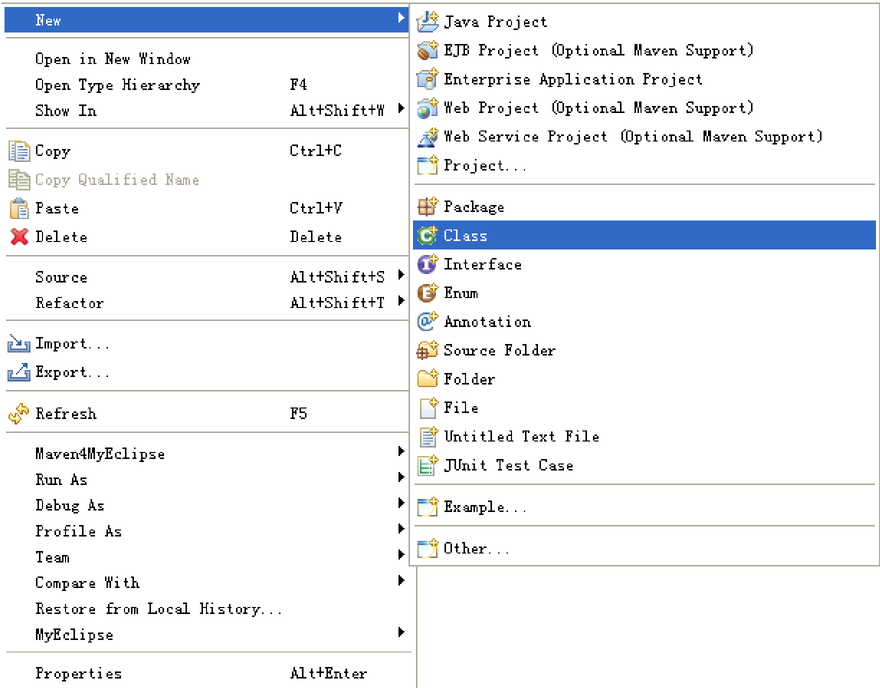


图 3–215

Java 类属性设定如图 3-216 所示。

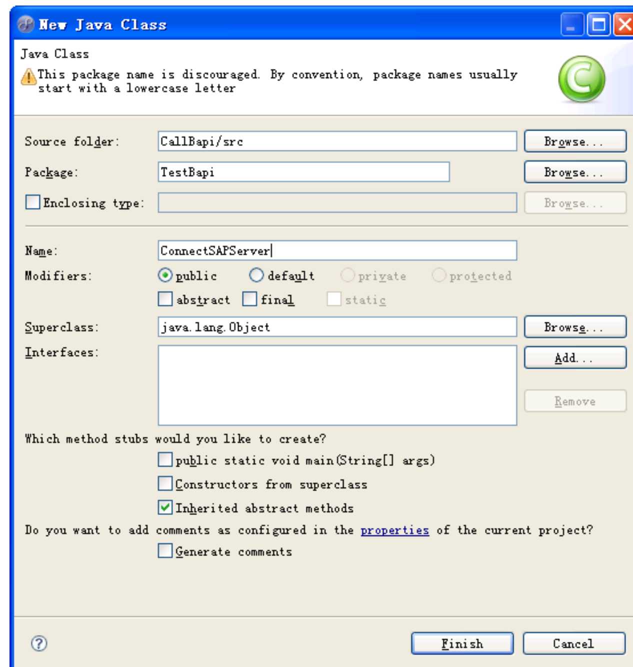


图 3-216

单击 Finish 按钮，编辑代码如下：

```
package TestBapi;
import com.sap.mw.jco.IFunctionTemplate;
import com.sap.mw.jco.JCO;
import java.util.Properties;
public class ConnectSAPServer {
    public static void main（String[] args） {

/*获得一个到 SAP 系统的连接    START    */
        Properties logonProperties = new Properties（）;
        logonProperties.put（"jco.client.ashost","192.168.90.12"）;    //系统的 IP 地址
        logonProperties.put（"jco.client.client","000"）;               //要登录的客户端
        logonProperties.put（"jco.client.sysnr","00"）;                 //系统编号
        logonProperties.put（"jco.client.user","bcuser"）;              //登录用户名
        logonProperties.put（"jco.client.passwd","minisap"）;           //用户登录口令
        //用上述条件创建一个连接对象
        JCO.Client myConnection = JCO.createClient（ logonProperties ）;

/*获得一个到 SAP 系统的连接    END      */
        myConnection.connect（）;           //进行实际连接
        //如果连接不为 null 并且处于活动状态
```

```
if （myConnection != null && myConnection.isAlive（））  {
    //从连接获得一个逻辑意义上的“仓库”对象（Repository）
    JCO.Repository myRepository =
        new JCO.Repository（"Repository",  //只是一个名字
                           myConnection）; //活动的连接
    //要调用的 SAP 函数名称
    String strFunc = "Z_BAPI_GET_TEST";
    //从“仓库”中获得一个指定函数名的函数模板
    IFunctionTemplate ft = myRepository.getFunctionTemplate（strFunc.toUpperCase（））;
    //从这个函数模板获得该 SAP 函数的对象
    JCO.Function function = ft.getFunction（）;
    //获得函数的 import 参数列表
    JCO.ParameterList input = function.getImportParameterList（）;

/*设定个别参数  START  */
    //设定一个 import 参数的值。参数名为“MAX_ROWS”，设定值为 10
    //input.setValue（10, "MAX_ROWS"）;

/*设定结构参数  START  */
    //如果参数是一个结构，用参数名获得一个对应类型的结构对象
    JCO.Structure sFrom = input.getStructure（"IMPARAM"）;
    //设定结构中的变量。
    sFrom.setValue（"1A", "CLASS"）;
    sFrom.setValue（"2009", "YEAR"）;
    //将处理好的结构赋给该参数
    input.setValue（sFrom, "IMPARAM"）;
/*设定结构参数  END    */
/*设定 table 参数  START  */
    //用参数名获得一个对应类型的内部表对象
    //JCO.Table tDateRange = function.getTableParameterList（）.getTable（"DATE_RANGE"）;

    //新增一条空行
    //tDateRange.appendRow（）;
    //定位到第 0 行
    //tDateRange.setRow（0）;
    //设定该行对应变量
    //tDateRange.setValue（"I", "SIGN"）;
    //tDateRange.setValue（"EQ", "OPTION"）;

    //tDateRange.setValue（"20040330", "LOW"）;

    //新增一条空行
    //tDateRange.appendRow（）;
    //定位到第 1 行
```

```
            //tDateRange.setRow（1）;
            //设定该行对应变量
            //tDateRange.setValue（"I", "SIGN"）;
            //tDateRange.setValue（"EQ", "OPTION"）;
            //tDateRange.setValue（"20040427", "LOW"）;
            //......
/*设定 table 参数    END        */
            //执行函数
            myConnection.execute（function）;

            //在执行函数后可用相同方式获得输出结果
            JCO.Table tests = function.getTableParameterList（）.getTable（"TESTDATA"）;
            //JCO.Table 对象可以直接输出到 html 文件
            tests.writeHTML（"C：/function.html"）;
            //也可以如下单独获得表中个别变量
            System.out.println（"班级" + "\t"
                    +   "学年" + "\t"
                    +   "学生番号" + "\t"
                    +   "姓名" + "\t"
                    +   "语文成绩" + "\t"
                    +   "数学成绩" + "\t"
                    +   "自然成绩" + "\t"
                    +   "思品成绩"）;
            for  （int i = 0; i < tests.getNumRows（）; i++）  {
                        tests.setRow（i）;
                System.out.println（tests.getString（"ZZCLASS"）  + "\t"
                                + tests.getString（"ZZYEAR"）  + "\t"
                                + tests.getString（"ZZSTDNO"）  + "\t"
                                + tests.getString（"ZZNAME"）  + "\t"
                                + tests.getDouble（"ZZTENKOK"）  + "\t"
                                + tests.getDouble（"ZZTENSAN"）  + "\t"
                                + tests.getDouble（"ZZTENRIK"）  + "\t"
                                + tests.getDouble（"ZZTENSHA"））;
            }
            //断开连接
            myConnection.disconnect（）;
        } else {
            System.out.println（false）;
        }
    }
}//：~
```

注：大多数的 JCO 包的版本密码会被变成大写后再发送到 SAP 处验证，因为较新版本的 SAP 对密码的大小写是敏感的。所以 SAP 用户的密码如果包含有字母的话，要注意密码的兼容性，要么全是数字，要么全是大写，才能成功连接。

步骤二：准备测试数据，如图 3-217 所示。

数据浏览器：表格 ZZTEST_V 选择条目　　10

表　　：　　　　ZZTEST_V
被显示的字段：　　8 的　　8　固定栏目：　　　　　　8　列表宽度　　0250

班级	学年	学生番号	语文成绩	数学成绩	自然成绩	思品成绩	姓名
1A	2009	00001	60	68	70	81	郑喜定
1A	2009	00002	62	65	74	83	魏喜奎
1A	2009	00003	62	65	71	89	金喜善
1A	2009	00004	61	63	75	88	洪喜官
1A	2009	00005	64	66	75	88	朴喜珍
1A	2009	00006	64	66	75	88	王喜凤
1A	2009	00007	64	66	76	88	刘喜仁
1A	2009	00008	62	66	72	88	白喜医
1A	2009	00009	62	63	72	82	赵喜庆
1A	2009	00010	62	63	74	85	张喜早

图 3-217

步骤三：编译并按以下路径运行程序，如图 3-218 所示。

图 3-218

运行结果如图 3-219 所示。

Problems　@ Javadoc　Declaration　Console　Packages

<terminated> ConnectSAPServer [Java Application] D:\MyEclipse 6.5\jre\bin\javaw.exe (Aug 18, 2011 3:54:35 PM)

班级	学年	学生番号	姓名	语文成绩	数学成绩	自然成绩	思品成绩
1A	2009	00001	郑喜定	60.0	68.0	70.0	81.0
1A	2009	00002	魏喜奎	62.0	65.0	74.0	83.0
1A	2009	00003	金喜善	62.0	65.0	71.0	89.0
1A	2009	00004	洪喜官	61.0	63.0	75.0	88.0
1A	2009	00005	朴喜珍	64.0	66.0	75.0	88.0
1A	2009	00006	王喜凤	64.0	66.0	75.0	88.0
1A	2009	00007	刘喜仁	64.0	66.0	76.0	88.0
1A	2009	00008	白喜医	62.0	66.0	72.0	88.0
1A	2009	00009	赵喜庆	62.0	63.0	72.0	82.0
1A	2009	00010	张喜早	62.0	63.0	74.0	85.0

图 3-219

生成 htm 文件内容如图 3–220 所示。

Table: ZBAPI_TEST_EXPORT No. of Rows: 10 Row-length: 45 (chars) 49 (bytes)								
Index:	1	2	3	4	5	6	7	8
Name:	ZZCLASS	ZZYEAR	ZZSTDNO	ZZNAME	ZZTENKOK	ZZTENSAN	ZZTENRIK	ZZTENSHA
Type:	CHAR	NUM	CHAR	CHAR	BCD	BCD	BCD	BCD
Size:	2	4	5	30	2	2	2	2
Offset:	0	2	6	11	41	43	45	47
Decimals:	0	0	0	0	0	0	0	0
1A	"1A"	2009	"00001"	"郑喜定"	60	68	70	81
2A	"1A"	2009	"00002"	"魏喜奎"	62	65	74	83
3A	"1A"	2009	"00003"	"金喜善"	62	65	71	89
4A	"1A"	2009	"00004"	"洪喜官"	61	63	75	88
5A	"1A"	2009	"00005"	"朴喜珍"	64	66	75	88
6A	"1A"	2009	"00006"	"王喜凤"	64	66	75	88
7A	"1A"	2009	"00007"	"刘喜仁"	64	66	76	88
8A	"1A"	2009	"00008"	"白喜医"	62	66	72	88
9A	"1A"	2009	"00009"	"赵喜庆"	62	63	72	82
10A	"1A"	2009	"00010"	"张喜早"	62	63	74	85

图 3–220

第 4 章　增　　强

增强（Enhancement）的概念其实很广，SAP 标准系统中的所有二次开发都可以理解为增强。做增强的原因是标准的 SAP 业务系统不能满足实际需求，所以在标准的系统基础上增加不同的功能来达到不同企业的要求，换句话说今天开发过程中所有的 ADD-ON 都可以说是增强开发。

但开发人员所说的增强主要是标准系统事先预留好的接口，根据不同业务需求来进行开发，称为增强，又叫出口（EXIT）。

4.1　概述

在 SAP 系统设计中，增强被定义为程序执行到最后必经的一段程序。为什么会有出口呢？因为正常的业务系统不能满足实际需求，这时可以在出口中增加一些功能，来达到要求。

在 SAP 中出口就是标准程序最后要调用的一个函数或者功能模块，这个函数或功能模块的输入已经由 SAP 定义好了。

用户写的 SAP 出口程序在系统升级的时候会被保留，而如果更改标准程序在 SAP 系统升级的时候则会被覆盖。

下面介绍一下什么时候需要使用出口以及出口的种类：

- 在业务检查时：比如在某个工厂发货，可以设定在某个库位的出货只能使用某种移动类型。
- 在需要界面增强时：比如用户对某个字段要求大写，但是最终用户不按规范操作，这也可以在出口中自动转换，有些模块甚至能自定义数据库字段，并且可以在出口中增加字段输入。还有的模块能对输入数据检查，甚至实现自动替代等功能。
- 有不规则业务时：比如按某种条件定价，可以设定从自己定义的表中按某种条件取值。
- 搜索帮助的出口，可以对 SAP 标准的搜索帮助做权限控制等。

4.1.1　第一代增强

SAP 的增强发展至今已有四代，第一代增强基于源代码，是 SAP 提供一个空代码的子过程。在这个子过程中用户可以添加自己的代码，控制自己的需求。这类增强都需要修改 SAP 的标准代码，它们在发布的时候都是空的，集中在一些文件名倒数第二个字符为 Z 的包含程序中。由于是在标准程序中，所以所有程序的全局数据都可以使用，但同时系统升级时会被新版本覆盖；这种源代码增强和屏幕增强的说明可以从事务码 SPRO 后台配置中相关模块的路径里面找到；一般是以 UserExit_打头的子模块，可形象地称其为用户出口，如图 4-1 所示。

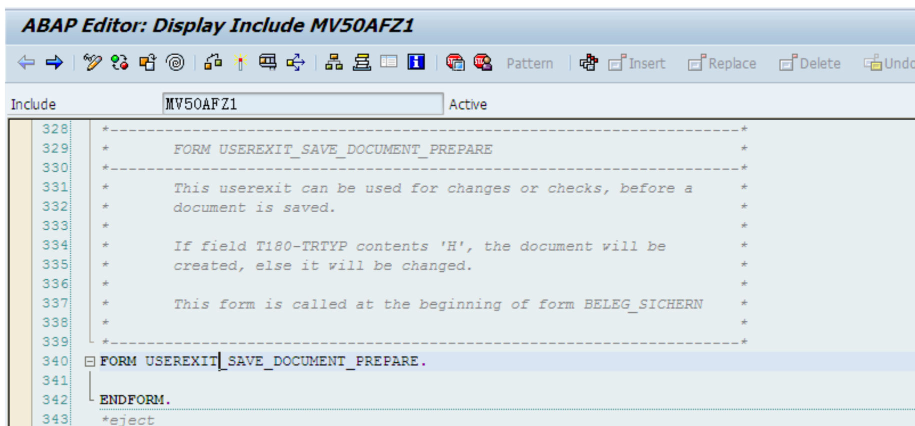


图 4-1

找到所要增强的程序，查找 UserExit_关键字。例如上图中 MV50AFZ1 里的 user_exit_save_document。大部分 SD 内的出口都是这种形式的。

查找此类出口的方法：

■ 在增强程序中查找第二个字符为 Z 的 Include 包含程序。

■ 在后台查找对应模块增强；在程序中搜索 USEREXIT_打头的关键字子程序；在 SAP 中以 PERFORM 的形式被调用。

■ 字节列出对象列表“Display Object List”按钮，在子程序列表下查找“UserExit”打头的子程序，如图 4-2 所示。

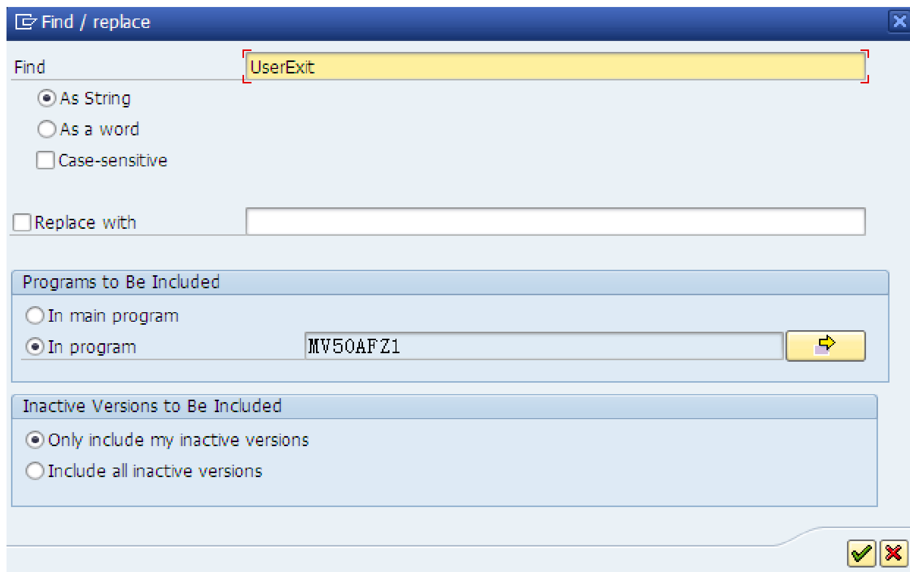


图 4-2

这类增强事先要到 SAP 的 Service Market Place 申请对象键（Access Key），然后才能修改子程序，如图 4-3 所示。

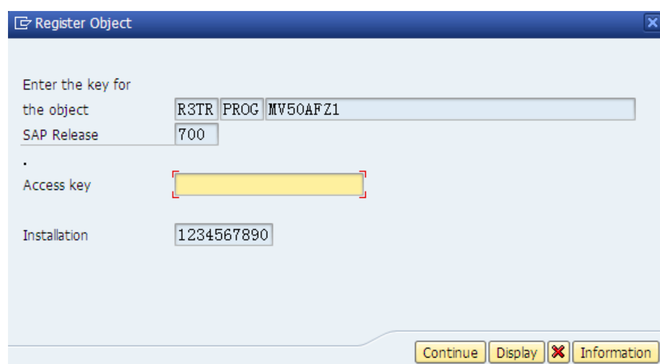


图 4-3

这种出口据说是第一代的用户出口，它们 Include 在 SAP 标准程序的源代码里，可以说它们是源代码的一部分，更改这种出口就相当于更改 SAP 标准程序，是需要申请对象键才能更改的，不需要建立工程。很多以 ZZ 或者 Z 结尾的 Include 标准程序里都有以 UserExit_开始的子程序，例如上面 mv50afz1 里的 user_exit_save_document。大部分 SD 的出口里，都是这种形式的（Form）。由于子程序中带有关键字 UserExit，故被称为 User Exits（用户出口）。与第二代增强相比，User Exit 只能在 SAP 应用程序中添加功能，不能定义自己的菜单、屏幕及关键字处理。

4.1.2 第二代增强

第二代增强（基于函数模块的增强）用事务 SAP 增强管理（事务代码：SMOD）和增强编辑器（事务代码：CMOD）维护。在 SAP 发布的版本中，使用 Call Customer-Function ‘xxx’调用函数模块，所以可以通过在程序中搜索 Customer-Function 来查找第二代增强，第二代增强函数名构成为 Exit_程序名_’ xxx（3 digital number）’。它们在发布的时候只有一句代码 include “zxxxxxxx”，修改时无需像第一代增强一样申请对象键（Access key），直接按两次〈Enter〉键就可以了。但同时这种增强以及后面几代的增强都不能像第一代一样随便使用程序的全局数据，只能使用接口中传递进来的参数。由于带有关键字 Customer，故被称为 Customer Exits（客户出口）。

第二代增强主要有 4 类：

- E. Ehancement exits：这些出口以 Exit_打头，可以在 SE37 中查看，也可以在数据字典 TFDIR（函数表）中查询 Exit_打头的函数。
- C.GUI codes（GUI 增强）：用于区域菜单和工具菜单栏的增强。
- S.Subscreens（屏幕增强）：用于子屏幕设计增强。
- T.include structure 增强：用于表结构的增强。

还有一种按照用途分类的标准：

- 功能模块出口（Function Module Exits）：在 SAP 标准事务中添加业务逻辑，一般用于验证标准事务中的字段。
- 屏幕出口 （Screen Exits）：定义自己的子屏幕用以嵌入应用屏幕，在标准事务中添加子屏幕。

■ 菜单出口（Menu Exits）：定义自己的菜单项或激活屏蔽标准事务中的菜单项。
■ 表出口（Table Exits）：与上节中的 T 型出口一样，属同一类型。
■ 数据元素出口（Data Element Exits）：在 ABAP/4 字典的关键字数据元素中添加文档。结果是在使用这些数据元素的字段处按〈F1〉键后会出现自定义的说明文档。
■ 关键字出口（Keyword Exits）：在 ABAP/4 字典中的关键字数据元素更改表示文本。
■ 字段出口（Field Exits）：根据 ABAP/4 字典中域验证某些标准事务的某些屏幕字段。

注：RZ10 配置系统参数，并重启应用服务器，如图 4-4 所示。

Display Profile 'ECC_DVEBMGS00_ECC6' Version '000001'

Parameter

2011/08/18 Active parameters 13:24:36

Parameter Name	Parameter value
INSTANCE_NAME	DVEBMGS00
DIR_CT_RUN	$(DIR_EXE_ROOT)\$(OS_UNICODE)\NTI386
DIR_EXECUTABLE	$(DIR_INSTANCE)\exe
jstartup/trimming_properties	off
jstartup/protocol	on
jstartup/vm/home	C:\j2sdk1.4.2_09
jstartup/max_caches	500
jstartup/release	700
jstartup/instance_properties	$(jstartup/j2ee_properties);$(jstartup/sdm_properties)
j2ee/dbdriver	$(DIR_EXECUTABLE)\ojdbc14.jar
PHYS_MEMSIZE	512
rdisp/wp_no_dia	6
rdisp/wp_no_btc	3
abap/fieldexit	YES

图 4-4

1．Enhancement比较重要的表

■ MODSAP：这个表里重要的字段增强名（Name）、组件类型（TYPE：ECST）、组件功能模块名（Member）里面记录了所有 Enhancement 的增强。
■ TFDIR：所有的函数表、重要字段 FuncName（函数名）、功能模块激活状态 MAND（如果是 C，代表此函数模块激活）。
■ TSDIR：动态程序区 CALL CUSTOMER SUBSCREEN（屏幕增强）。
■ CUATEXTS：修改 GVI 界面，菜单文本被客户改变（GUI 菜单文本增强）。
■ MODSAPA：SAP 扩展的属性。
■ MODATTR：SAP 增强项目属性。
■ TADIR：资源库对象的目录。
■ TFTIT：函数功能的短文本。

2．查找Enhancement的方法

（1）查找关键字

在程序中搜索 Customer-function，找到后面的 3 位数字后缀，然后出口函数组成就是 Exit_程序名_3 位数字后缀。

（2）代码查找增强

利用如下代码查找 Enhancement（有些 Exit 并不能找到）：

```
*---------------------------------------------------------------------*
*
```

```
* Program ID              ：ZENHANCEMENT_FIND_EXIT
* Program Description ：增强点查询
* Create   User           ：BCUSER
* Create   Date           ：2011/02/20
*
*-------------------------------------------------------------------*
* Function Overview
*  程序实现以事务代码为参照进行交互式查询.
*  技术点：动态下载报表，结构体嵌套
*
*-------------------------------------------------------------------*
* Change List             ：
*      change NO.   Change Date    Change User    Change Detail
*      0000001       2011/02/20     BCUSER          创建程序
*-------------------------------------------------------------------*
REPORT   ZENHANCEMENT_FIND_EXIT
           NO STANDARD PAGE HEADING.
*-------------------------------------------------------------------*
* TABLES：
*-------------------------------------------------------------------*
TABLES ：  TSTC,                    "SAP  事务代码
           TADIR,                   "资源库对象的目录
           MODSAPT,                 "SAP 增强-短文件
           MODACT,                  "修正
           TRDIR,                   "系统表 TRDIR
           TFDIR,                   "功能模块
           ENLFDIR.                 "功能模块的附加属性
TABLES ：  TSTCT.                   "事务代码文本
*-------------------------------------------------------------------*
* DATA
*-------------------------------------------------------------------*
*  资源库对象的目录内表
DATA ：  JTAB LIKE TADIR OCCURS 0 WITH HEADER LINE.
DATA ：  CURSOR_FIELD（30）.          "CURSOR 字段
*  传输组织器的开发类
DATA ：  V_DEVCLASS LIKE TADIR-DEVCLASS.
*-------------------------------------------------------------------*
* PARAMETERS
*-------------------------------------------------------------------*
*  事务代码
PARAMETERS ：  P_TCODE LIKE TSTC-TCODE OBLIGATORY.
*-------------------------------------------------------------------*
*START-OF-SELECTION
*-------------------------------------------------------------------*
START-OF-SELECTION.
*  事务代码取得
```

```
  SELECT SINGLE * FROM TSTC WHERE TCODE EQ P_TCODE.
* 成功的场合
  IF SY-SUBRC EQ 0.
*   资源库对象的目录取得
    SELECT SINGLE * FROM TADIR
        WHERE PGMID    = 'R3TR'     "请求和任务中的程序标识
        AND   OBJECT   = 'PROG'     "对象类型
        AND   OBJ_NAME = TSTC-PGMNA."对象目录中的对象名
    MOVE ：  TADIR-DEVCLASS TO V_DEVCLASS.
*   失败场合
*     系统表 TRDIR 取得
      SELECT SINGLE * FROM TRDIR
          WHERE NAME = TSTC-PGMNA.
*     程序类型为函数组的场合
      IF TRDIR-SUBC EQ 'F'.
*       功能模块取得
        SELECT SINGLE * FROM TFDIR
            WHERE PNAME = TSTC-PGMNA.
*       功能模块的附加属性取得
        SELECT SINGLE * FROM ENLFDIR
            WHERE FUNCNAME = TFDIR-FUNCNAME.
*       资源库对象的目录取得
        SELECT SINGLE * FROM TADIR
            WHERE PGMID    = 'R3TR'
            AND   OBJECT   = 'FUGR'
            AND   OBJ_NAME = ENLFDIR-AREA.

        MOVE ：  TADIR-DEVCLASS TO V_DEVCLASS.
      ENDIF.
    ENDIF.
*   资源库对象的目录取得
    SELECT * FROM TADIR
        INTO TABLE JTAB
        WHERE PGMID   = 'R3TR'
        AND OBJECT    = 'SMOD'
        AND DEVCLASS = V_DEVCLASS.
*   事务代码文本
    SELECT SINGLE * FROM TSTCT
        WHERE SPRSL EQ SY-LANGU
        AND   TCODE EQ P_TCODE.

    FORMAT COLOR COL_POSITIVE INTENSIFIED OFF.
*   输出事务代码
```

```
    WRITE：/（19） '事务代码：',  20（20） P_TCODE,  45（50） TSTCT-TTEXT.
    SKIP.
*   增强信息不为空的场合
    IF NOT JTAB[] IS INITIAL.
      WRITE：/（95） SY-ULINE.
      FORMAT COLOR COL_HEADING INTENSIFIED ON.
*       增强列表表头
      WRITE：/1 SY-VLINE,
      2 '增强名',
      21 SY-VLINE ,
      22 '描述',
      95 SY-VLINE.
*       增强列表明细
      WRITE：/（95） SY-ULINE.
      LOOP AT JTAB.
        SELECT SINGLE * FROM MODSAPT
        WHERE SPRSL = SY-LANGU AND
        NAME = JTAB-OBJ_NAME.
        FORMAT COLOR COL_NORMAL INTENSIFIED OFF.
        WRITE：/1 SY-VLINE,
        2 JTAB-OBJ_NAME HOTSPOT ON,
        21 SY-VLINE ,
        22 MODSAPT-MODTEXT,
        95 SY-VLINE.
      ENDLOOP.
*       增强列表表尾
      DESCRIBE TABLE JTAB.
      SKIP.
      FORMAT COLOR COL_TOTAL INTENSIFIED ON.
      WRITE：/ '增强个数：' , SY-TFILL.
*   增强信息为空的场合
    ELSE.
      FORMAT COLOR COL_NEGATIVE INTENSIFIED ON.
      WRITE：/（95） '没有用户增强。'.
    ENDIF.
* 失败的场合
  ELSE.
    FORMAT COLOR COL_NEGATIVE INTENSIFIED ON.
    WRITE：/（95） '事务代码不存在！'.
  ENDIF.
*---------------------------------------------------------------------*
* AT LINE-SELECTION
*---------------------------------------------------------------------*
```

```
AT LINE-SELECTION.
* 取得光标字段
  GET CURSOR FIELD CURSOR_FIELD.
  CHECK CURSOR_FIELD（4） EQ 'JTAB'.
  SET PARAMETER ID 'MON' FIELD SY-LISEL+1（10）.
`  CALL TRANSACTION 'SMOD' AND SKIP FIRST SCREEN.
```

（3）利用系统函数寻找增强

常与出口相关的函数有：

- DYNP_VALUES_READ。
- MODX_ALL_ACTIVE_MENUENTRIES（菜单增强）。
- MODX_FUNCTION_ACTIVE_CHECK（出口函数增强）。
- MODX_MENUENTRY_ACTIVE_CHECK（菜单增强）。
- MODX_SUBSCREEN_ACTIVE_CHECK（屏幕增强）。

可以在函数最后设置断点，然后通过运行对应的增强程序就会调用这样的函数，这样就可以在调试模式下找到对应的增强。

注：出口的应用要在激活的状态下，也就是在 TFDIR 下的 MAND 字段为 C。激活的时候要同时激活程序和出口函数。一般情况下，知道出口模块可以找到对应的增强，如果未激活状态需要将其激活。

4.1.3 第三代增强

BADI 是一个新的 SAP 增强技术，是基于 ABAP 的面向对象技术。它们可以被插入到 SAP 系统，以适应用户由于行业的不同需要不同的功能以满足具体要求（包括在标准交付）。SAP 让客户在软件中预先确定这些点。

基于面向对象概念的增强 BADI（Business Add-in），源代码的发布是通过接口的方法调用来实现使用的。用户增强实际上是实现一个或多个基于这个接口的实现类，因为接口类实际上是一个抽象类，所以对同一个增强会出现不同的源代码，这些不同的源代码是通过过滤器（Adapter）来区别用于不同的业务场景的。这种增强是用事务 SE18 和 SE19 来实现的。

SAP 预定义了一些接口（Interface），客户可以自行定义实现 Interface 的类（Class），在标准程序中会使用调用客户自定义类（Class）的实例（Instance），获取实例的过程使用了工厂模式，因此获取过程对用户是未知的。不过 BADI 也有一些局限性，就是有一些 Interface 是固定的，有的时候会发现需要 Mix 客户化代码的时候找不到 Interface。

业务交易事件（BTE，Business Transaction Events）是 SAP 的一种增强方式，技术上讲是介于第二代增强（Customer Exits）和 BAdIs 之间的产物。

BTE（Business Transaction Events）为会计模块独有的增强，这种增强用于财务会计模块（Open FI）、总账会计（FI- GL）、应收账款和应付账款（FI-AR/FI-AP）及销售和分销（SD）的组件。

BTE 增强有 2 种类型，类似于会计凭证验证和替代。

- Publish and Subscribe Interface：只提供 SAP 数据源，可以供外部程序使用或者达到数

据检查的目的。

用于告诉外部软件在 SAP 标准应用中触发了某些事件，并且把产生的数据提供给它们。外部软件不能给 R/3 系统返回任何数据，可以用在以下地方。

- 主数据被创建、修改或锁定。
- 凭证被输入、预制、修改或冲销。
- 项目被清除或重置。

也可以在附加的开发中基于此事件和数据产生附加的处理：

- 启动工作流。
- 产生或修改附加数据。
- 请求通信。

■ Process Interface：可以达到数据修改的目的，用来增强标准的业务流程。

用于控制与标准 R/3 系统处理方法不同的业务流程，也就是用 Process Interface 替换标准处理。它们干预标准流程，并返回数据给 SAP 应用。

1．第二代增强（Customer exits）和BTE的不同之处

Customer exits 只有 SAP 和 Customer 两方参与流程，SAP 提供软件交付给最终客户，客户可以用 Customer exits 对标准功能进行增强；BTE 则更面向组件，有多方参与流程。流程中所有的参与者都可能是增强的用户和提供者。由于 Customer exits 只能使用一次，不能满足需求，因此 SAP 在 4.0 版本中开发了新的增强技术，并允许重复使用。

Open FI 基于以下逻辑，应用程序开发者必须定义某个函数的接口。应用时，一个分配用的表被读取，被分配的程序被自动调用。

Customer exits 这种技术不同于其他增强的是，它只允许有一个实施的增强，但是可以按照需求和一定顺序调用多个已实现的增强。可以特定行业和特定国家增强，这可以很容易地通过/定制业务的需求，其中 SAP 不提供标准的功能，只能通过定制或改进。

BTE 即 OPEN FI 此功能主要是用于 FI 催款和信用管理，例如在 SAP 的标准催款功能中，根据 Partner Function 发送付款通知，以 Partner 为基础，根据原始凭证的金额，可以发送相同的通知文档。

2．BADI和BTE的不同之处

BTE 同 BADI 的技术实现基本上是一致的，然而，这两个增强技术各不相同，体现在以下几点：

■ Open FI 仅限于代码增强，而不能像 BADI 一样对用户界面进行增强。

■ Open FI 只会在 3 个层面上进行（SAP—合作伙伴—客户），而 BADI 加载项可以实现各个软件层的增强。

■ Open FI 使用函数进行增强，而 BADI 则使用面向对象实例化的方法进行增强。

3．BADI的查找方法

BADI 的查找方法有以下几种：

■ BADI 对象的信息存储在 SXS_INTER、SXC_EXIT、SXC_CLASS 和 SXC_ATTR 这 4 个表中（参见 SECE 包）。

■ SAP BADI 程序都会调用 cl_exithandler=>get_instance 来判断对象是否存在，并返回实例，这个方法实际就是对上面 4 个表以及视图 V_EXT_IMP 和 V_EXT_ACT 进行查询

和搜索的。可以在这个方法处设置断点，获取实例。

- 它的调用方式是 call method（instance），可以通过 exit_handler 关键词来查找。
- 基于以上机理使用性能分析器（事务代码 ST05）选择“table buffer trace”而不是常用的“SQL trace”，然后查找上面的几个表和视图找到对应的 BADI。
- 使用 SE24 进入 Class Builder，查看对象 CL_EXITHANDLER，进入其 get_instance 方法，为 get_class_name_by_interface 设置断点。Debug 需要增强的事务，在断点处查看 class_name 即可找到对应的 BADI。
- se18 查找接口，se19 实现接口就可以实现用户增强。

4．BTE的查找方法

BTE 的查找方法有以下两种：

- 运行事务码（例如 XK02），选择“System→Status”，双击“Program (screen)”，选中“In main program”，查找“OPEN_FI_PERFORM”字符串。
- IMG→财务会计全局设置(新)→工具→客户增强→业务事务事件(FIBF)，选择“环境”。

4.1.4 第四代增强

第四代其实是第三代的加强 Switch Framework。当 SAP 进入 NetWeaver 7.0 以后推出的新增强体系，将 BADI 进行了改进，叫新 BADI。还新增了 Enhancement Spot、Enhancement Section 以及隐式增强点的概念，基本可以在面向对象的程序里实现处处皆可增强的最高境界。

SAP 应用 Switch Framework 的主要目的是采用标准系统中的一个或多个行业的解决方案，以简化基于 ABAP 的系统框架。Switch Framework 允许客户通过使用交换机框架，对所有的行业解决方案中的资源库对象或对象的组件的可见性进行外部控制。客户不再需要安装一个行业解决方案，可以在需要时激活相关组件。

在程序中查找字符串 Enhancement，即可以在指定的位置进行增强。

4.2 第一代增强的应用

User Exits 是 SAP 第一代增强技术，其技术原理就是在程序中预先放置一个 Form...EndForm 的子程序，并在事物运行时 Call 这个 Form。如果这个 Form 是空的，则不执行任何操作，否则，执行相应的操作。User Exits 的写法和修改标准程序是一致的，要向 SAP 申请 Access Key 才能对这些程序进行修改。它与修改标准程序的区别在于在 SAP 做升级的时候，User Exits 会自动保留，标准程序如果不做特殊处理，就会被覆盖。

在做这样的增强时，一般把所要添加的代码都写在以 Z 开头的函数中，然后在 Form 中调用这个函数。由于该方法作用有限，本书不做举例介绍。

4.3 第二代增强的应用

4.3.1 数据元素及关键字

可以在 CMOD->GoTo->Text enhancements->Data elements/Keywords 路径中做字典文档的增强，如图 4-5 所示。

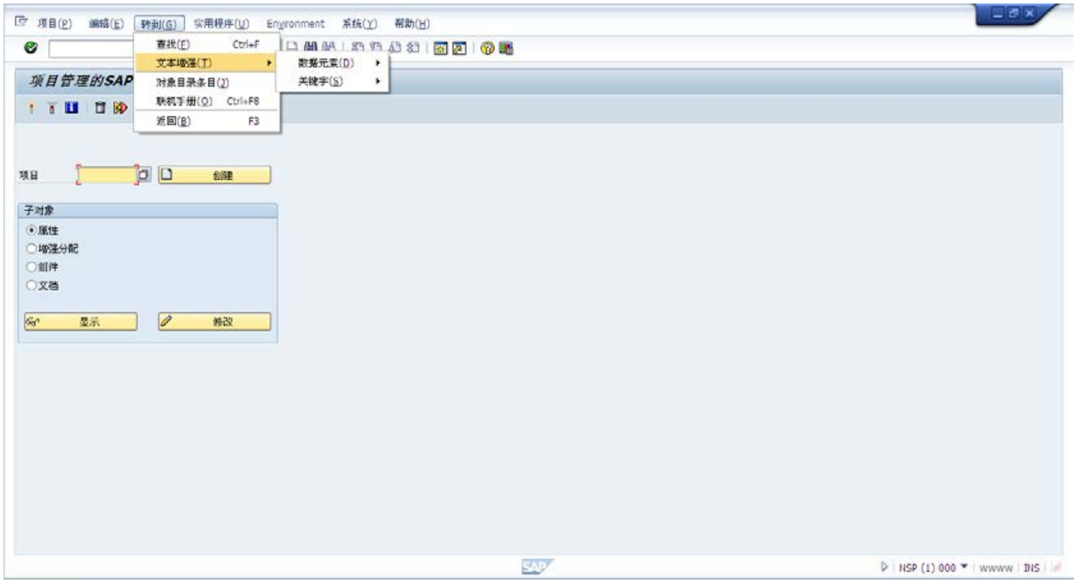


图 4-5

数据元素用以增强与其关联的文档帮助，关键字用于增强数据元素关联的文本标签。由于客户很少在这个层面上弃用 SAP 的标准功能，作用也比较有限，这里不做举例说明。

4.3.2 字段增强

字段增强技术基于数据元素（Data Element），如果屏幕的输入框参考了该数据元素，定义激活字段增强（Field Exit）后，程序响应一块函数代码，可以触发该字段增强逻辑。

技术上来讲有点像 Customer Exits，两者都是通过 Function 来实现出口的，但是其原理和用途又有很大不同，而且 Field Exits SAP 在 4.6 之后就不再支持了。因此如果要在后续版本中使用，需要增加系统参数 abap/fieldexit = YES，然后重新启动 SAP。

- 可以在 SE38 里运行程序 RSMODPRF 来寻找已经存在的 Field Exits，同时也可以进入 Field Exits 的管理界面。
- 可以在 CMOD 界面中输入事务 PRFB，也可以进入 Field Exits 的管理界面。

新建一个 Field Exits 时，只要有相应的 Data Element 在数据库中存在，就能建立出针对这个 Data Element 的 Field Exits。同时会生成一个 FIELD_EXIT_开头的 Function，用以做相应的增强。里面有两个参数，Input 和 Output，这两个参数是固定的，不能修改。Input 是将屏幕字段的值传入出口程序，而 Output 则是传出。

例如，对用户进行维护（SU01），只允许 SAP* 和 DDIC 实现用户的维护，其他用户没有权限进行维护。

步骤一：找出用户这个输入字段的数据元素，如图 4-6 所示。

步骤二：执行程序 RSMODPRF，不输入任何参数，进入字段增强的主界面，如图 4-7 所示。



图 4-6

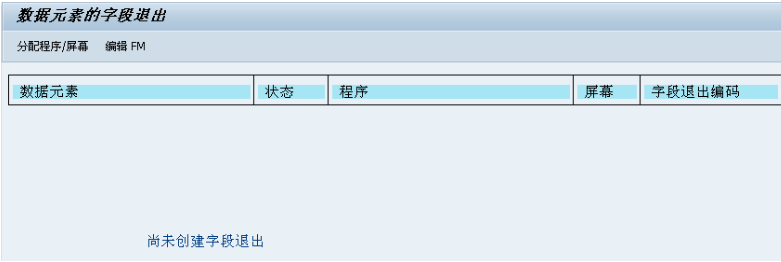


图 4-7

步骤三：按照以下路径创建新的字段增强，如图 4-8 所示。

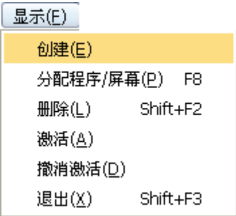


图 4-8

输入要增强的数据元素 XUBNAME，如图 4-9 所示。

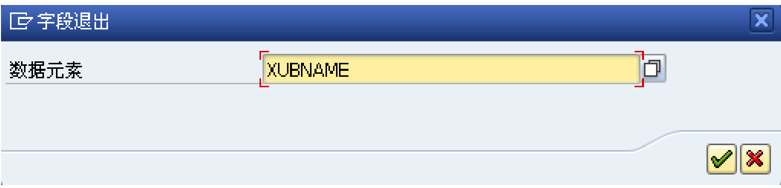


图 4-9

系统会自动创建一个以 FIELD_EXIT_开头的函数，用以做相应的增强，如图 4-10 所示。

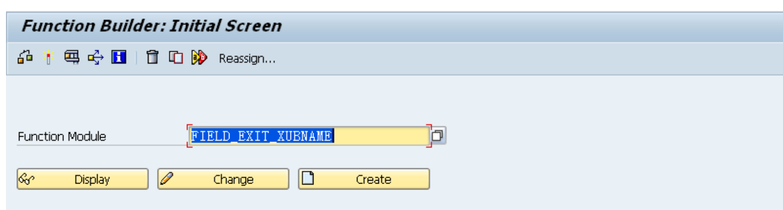


图 4-10

单击 Create 按钮，如图 4-11 所示。

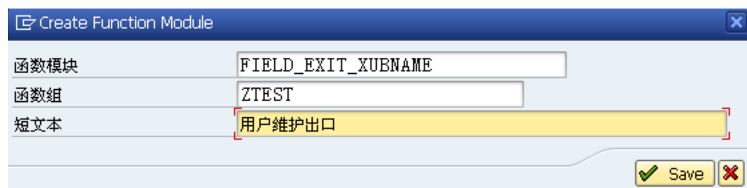


图 4-11

输入函数组及描述，单击 Save 按钮。

```
FUNCTION FIELD_EXIT_XUBNAME.
*"----------------------------------------------------------------------
*"*"Local interface:
*"    IMPORTING
*"        REFERENCE(INPUT)
*"    EXPORTING
*"        REFERENCE(OUTPUT)
*"----------------------------------------------------------------------
  DATA: M_STRING TYPE STRING.

  CONCATENATE '用户名:'   SY-UNAME     '无维护权限！ '
         INTO M_STRING.
  IF SY-UNAME <> 'YIDATEC' AND   SY-UNAME <> 'DDIC' AND   SY-UNAME <> 'SAP*' .
    MESSAGE M_STRING   TYPE 'A'.
  ENDIF.

ENDFUNCTION.
```

单击标准工具栏上的“保存”按钮，如图 4-12 所示。



图 4-12

单击标准工具栏上的“检查”按钮，如图 4-13 所示。



图 4-13

单击标准工具栏上的“检查”按钮，如图 4–14 所示。



图 4–14

单击标准工具栏上的检查按钮退回至管理画面，如图 4–15 所示。

数据元素的字段退出

分配程序/屏幕　编辑 FM

数据元素	状态	程序	屏幕	字段退出编码
XUBNAME	不活动的	全局的		

图 4–15

步骤四：激活新建的字段增强。

选中新建的字段增强，按照以下路径激活，如图 4–16 所示，效果如图 4–17 所示。

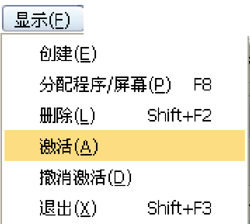


图 4–16

数据元素的字段退出

分配程序/屏幕　编辑 FM

数据元素	状态	程序	屏幕	字段退出编码
XUBNAME	活动的	全局的		

图 4–17

注：上述字段增强是对所有程序的增强，只要程序用到数据元素 XUBNAME，那么都会触发。在这里单击 分配程序/屏幕 按钮，可以将增强绑定到某个程序，如图 4–18 所示。



图 4–18

步骤五：测试新建的字段增强，如图 4–19 所示。



图 4–19

此类出口程序中，以下 ABAP 语句是受限制的：

- CALL SCREEN、CALL DIALOG、CALL TRANSACTION、SUBMIT。
- COMMIT WORK、ROLLBACK WORK。
- COMMUNICATION RECEIVE。
- EXIT FROM STEP-LOOP。
- MESSAGE I、MESSAGE W。
- STOP、REJECT。

所以也给出口的功能带来了限制。同时，调试在 Field Exit 会被忽略，这对纠错来说也是一个不利因素，只能做单元测试，而不能做集成测试。要将 Field Exits 做得更细，那就要给它分配程序及其对应的屏幕号。

4.3.3 GuiXT增强

在一个事务中，事务变式和屏幕变式是可以一起做的。当制作事务变式的时候，默认也会做一个事务初始屏幕的变式；但是想改变屏幕的布局，就要用到 GuiXT 了。

GuiXT 是一个第三方的软件，可以用来对 SAP 的界面进行修改，包括登录界面。

实例：事务 SE11 用来维护数据字典，现在需要一与之相仿的事务来维护数据库表。

步骤一：处理变式的维护（第一画面）。

通过事务代码 SHD0 进入事务和屏幕变式创建工具，如图 4–20 所示。

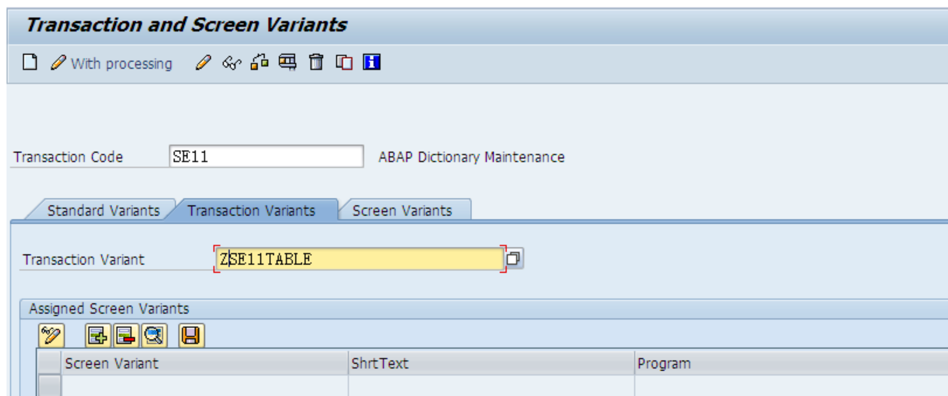


图 4–20

输入 Transaction Code ：SE11，Transaction Variant：ZSE11TABLE，然后单击工具栏中的按钮或者按〈F5〉键。

注：此处 Transaction Variant 变式名称必须以 Z 或 Y 开头。

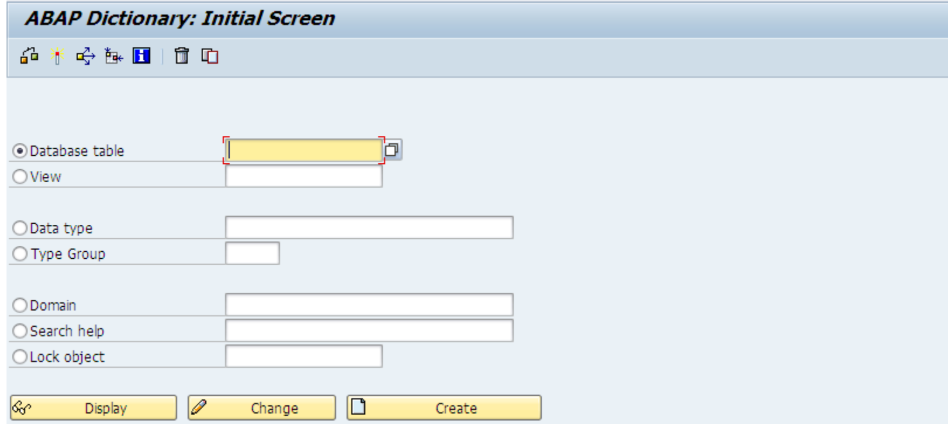


图 4-21

步骤二：处理变式的维护（第二画面）。

按〈Enter〉键弹出如图 4-22 所示的窗口将“Database table”以下的输入项隐藏。

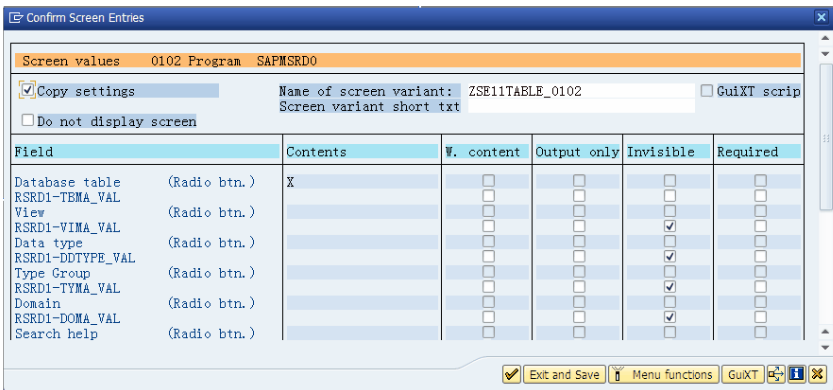


图 4-22

注：Menu functions用于屏蔽工具栏中的功能菜单。上述设定是将“Database table”以外的选项去掉，而选项所在文本需将 GuiTX Scripts 隐藏。

步骤三：处理变式的维护（第二画面）。

单击右下角的GuiXT按钮，弹出脚本编辑窗口，填写代码隐藏文本如图 4-23 所示。

注：上述代码是将“Database table”以外的文本去掉，并改变三个按钮的位置，具体应用请参照 GuiTX Scripts 相关资料。

单击按钮或按〈Enter〉键。

脚本编辑完并确定后，右上角的 GuiXT script 已经被勾选，如图 4-24 所示。

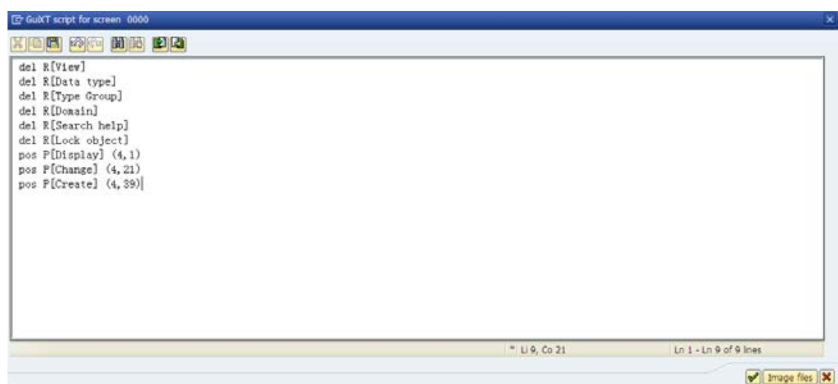


图 4–23

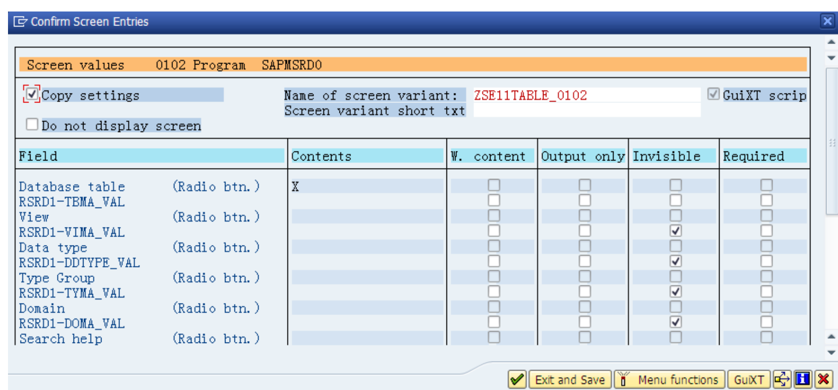


图 4–24

步骤四：处理变式的维护（第二画面）。

单击右下角的 Exit and Save 按钮，保存 GuiTX，如图 4–25 所示。

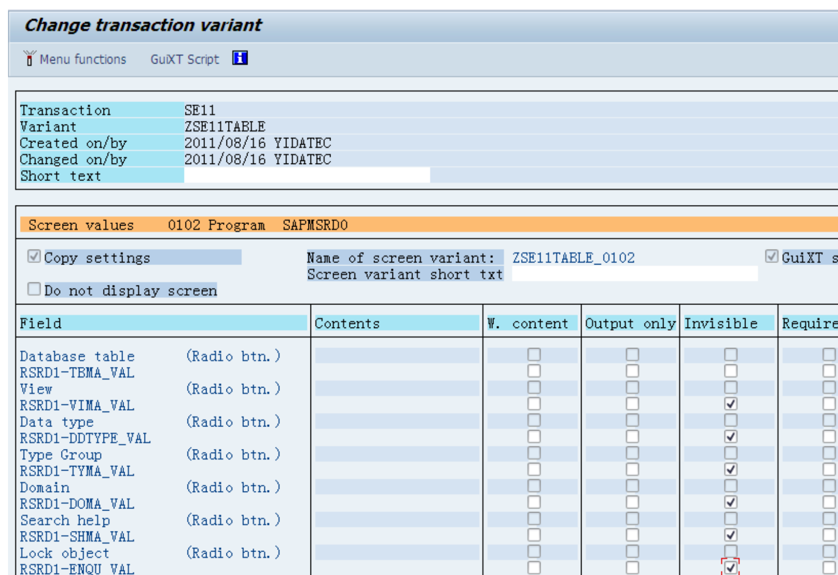


图 4–25

步骤五：处理变式的维护（第二画面）。

单击标准工具栏中的“保存”按钮，保存处理变式，如图 4-26 所示。

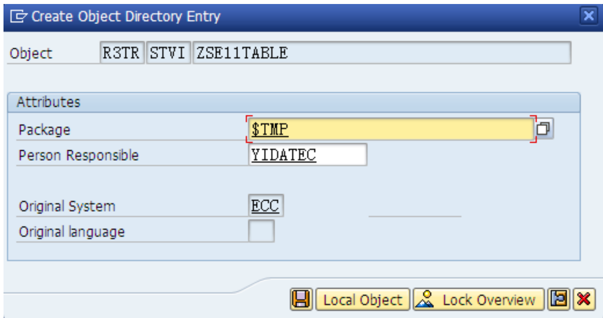


图 4-26

单击上图中的“保存”按钮，效果如图 4-27 所示。



图 4-27

按〈F3〉键返回到事务屏幕变式创建屏幕，如图 4-28 所示。

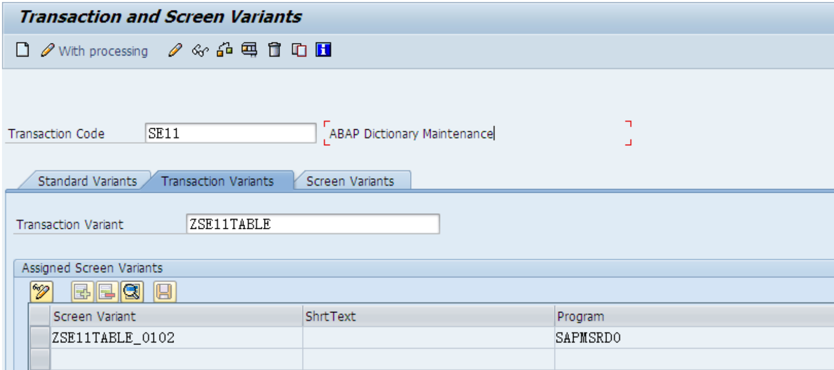


图 4-28

步骤六：处理变式的维护（第一画面）。

按照以下路径创建变式事务，如图 4-29 所示。

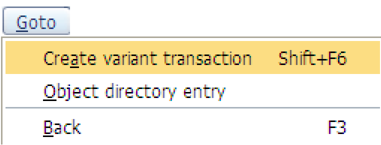


图 4-29

编辑事务如图 4-30 所示，单击按钮。

步骤七：处理变式的维护（第一画面）。

编辑处理事务如图 4-31 所示。

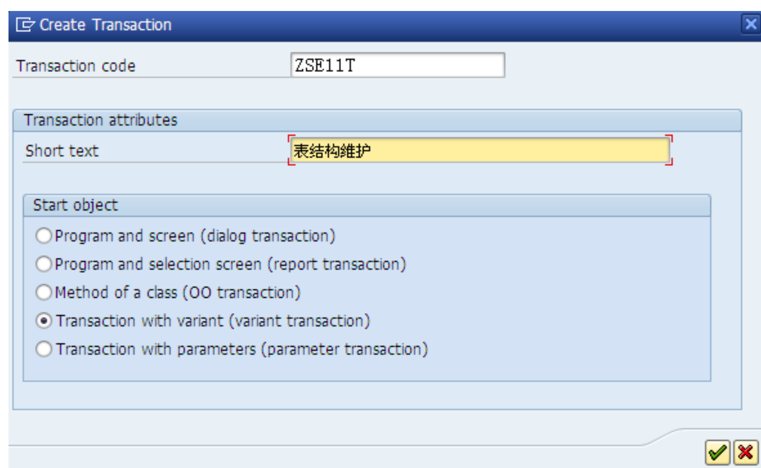


图 4–30

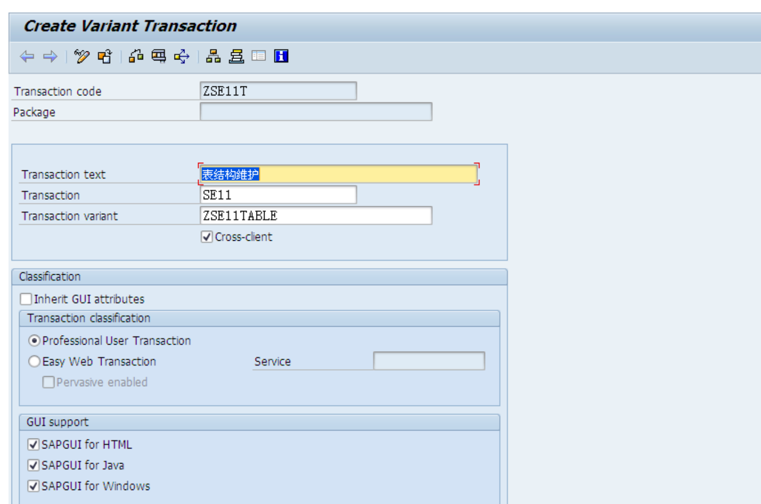


图 4–31

单击图 4–31 中的“保存”按钮，如图 4–32 所示。

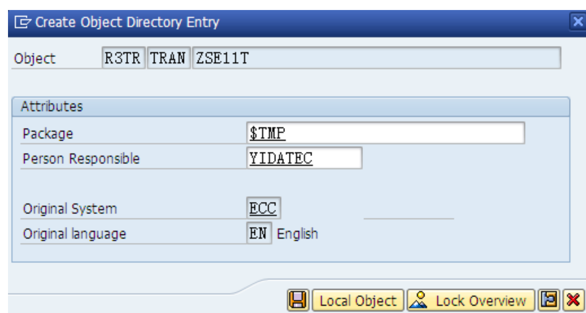


图 4–32

单击图 4–32 中的“保存”按钮，效果如图 4–33 所示。

Transaction code ZSE11T saved

图 4-33

步骤八：测试事务变式。

在命令栏中输入事务 SE11，效果如图 4-34 所示。

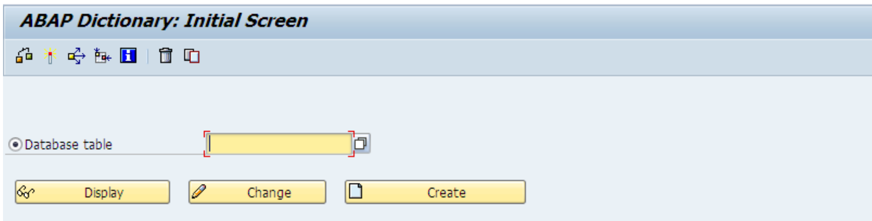


图 4-34

4.3.4　表增强

对于透明表及结构的增强，主要有两种，一种是 Append Structure，另一种就是 Include 形式。

Append Structure 可以扩展原来的表结构，但并不是在原来的表上加字段，而是再建一个附加的结构，物理上，数据是保存在两处的。而且，一个 Append Structure 只能有一张表结构。如果把有 Append Structure 结构的表进行复制的话，那么在新的表中，原有的 Append Structure 就变成了固定字段，物理表也变成了一张。

Include 格式多见于由于增强而自动生成的表字段，比如做屏幕增强的字段。当然，也可以自定义。相比 Append Structure，Include 可以包含多张表结构（Include 的 Data Elements 可以 Append Structure）。它不具有 Append Structure 的复制性能，即使复制，还是会以原结构存在于新表中。

做完增强后，如果想所加字段也出现在 SM30 表维护中，则需要通过以下路径生成表维护程序：SE11→进入相应表，并做完增强→Utilities→Table Maintenance Generator。

如果要反复更改增强，比如改变某一字段的长度，那么在减少字段时，需用 SE14 进行物理表的修改，否则无法激活。

实例：透明表 SFLIGHT08 是航班信息表，现在想将飞行员及误餐的信息加进去，增强前表结构如图 4-35 所示。

Dictionary: Display Table

Technical Settings　Indexes...　Append Structure...

Transp. Table　SFLIGHT08　Active

Short Description　Flight demo table

Attributes　Delivery and Maintenance　Fields　Entry help/check　Currency/Quantity Fields

Srch Help　Predefined Type

Field	Key	Initi...	Data element	Data Ty...	Length	Decim...	Short Description
PLANETYPE	☐	☑	S_PLANETYE	CHAR	10	0	Aircraft Type
SEATSMAX	☐	☑	S_SEATSMAX	INT4	10	0	Maximum capacity in economy class
SEATSOCC	☐	☑	S_SEATSOCC	INT4	10	0	Occupied seats in economy class
PAYMENTSUM	☐	☑	S_SUM	CURR	17	2	Total of current bookings
SEATSMAX_B	☐	☐	S_SMAX_B	INT4	10	0	Maximum capacity in business class
SEATSOCC_B	☐	☐	S_SOCC_B	INT4	10	0	Occupied seats in business class
SEATSMAX_F	☐	☐	S_SMAX_F	INT4	10	0	Maximum capacity in first class
SEATSOCC_F	☐	☐	S_SOCC_F	INT4	10	0	Occupied seats in first class

图 4-35

步骤一：ABAP 字典（编辑画面）。

显示 SAP 标准表 SFLIGHT08，单击应用工具栏中的Append-Struktur...按钮，如图 4-36 所示。



图 4-36

单击上图对话框中的按钮，弹出如下对话框，如图 4-37 所示。

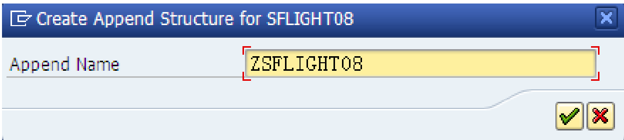


图 4-37

在上图中“Append Name”栏中输入构造体“ZSFLIGHT08”，单击对话框中的按钮。

步骤二：ABAP 字典（编辑画面）。

编辑构造体 ZSFLIGHT08，如图 4-38 所示。

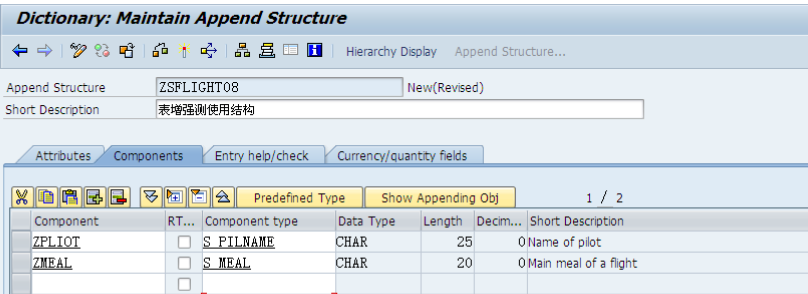


图 4-38

注：构造体 ID 必须以 Z 或 Y 作为头字母。

按照以下菜单路径将其激活，如图 4-39 所示。

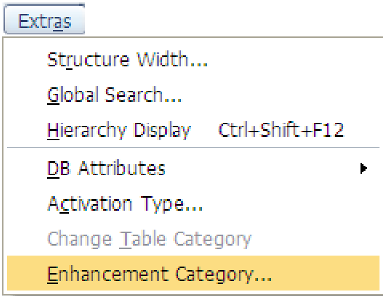


图 4-39

弹出以下对话框，如图 4-40 所示。

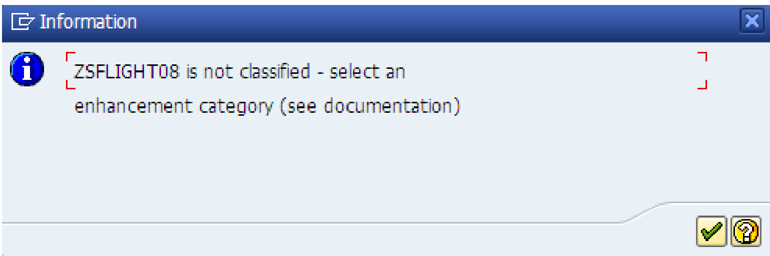


图 4-40

单击上图中的按钮，如图 4-41 所示。

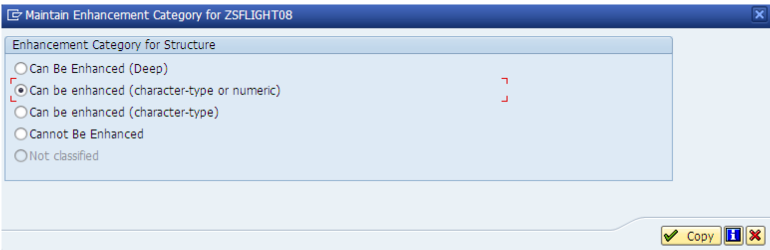


图 4-41

选中单选按钮“Can be enhanced（character-type or numeric）”选项并单击 Copy 按钮。

注：各选项说明如下：

Cannot be enhanced：该结构必须不被增强。

Can be enhanced or character type：所有的结构组件和它们的增强必须是字符类型（C、N、D、T）。包括原有的结构和所有增强功能，通过自定义或附加结构受这个限制。

Can be enhanced or character-type or numeric：结构和它的增强必须不包含任何深度的数据类型（表格、参考文献、字符串）。

Can be enhanced in any way：结构和它的增强可能包含的数据类型可以是任何类型的组件。

Not classified：该类别可以选择，例如对于一个过渡状态。但是，它不能被选择用于创建结构。

步骤三：ABAP 字典（编辑画面：保存、检查及有效化）。

单击标准工具栏中的“保存”按钮，如图 4-42 所示。

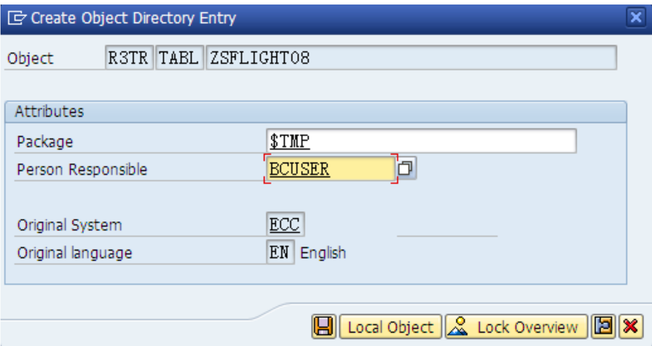


图 4-42

单击上图中的“保存”按钮，效果如图 4-43 所示。



图 4-43

单击应用工具栏中的“检查”按钮，如图 4-44 所示。

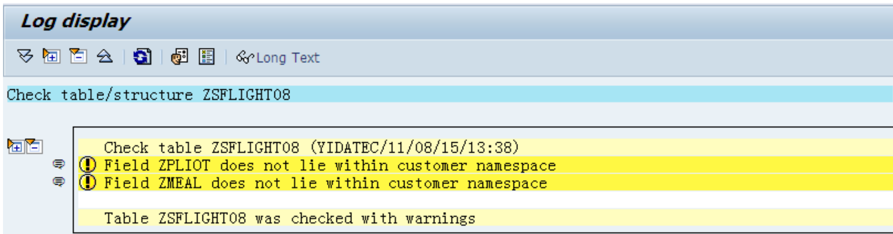


图 4-44

单击标准工具栏中的“有效化”按钮，如图 4-45 所示。

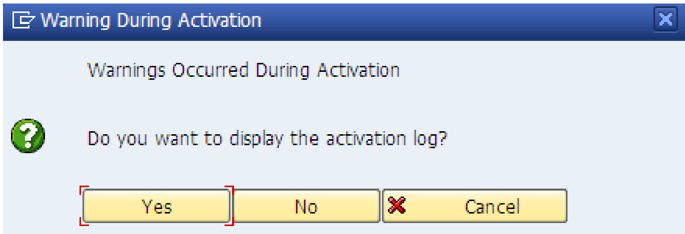


图 4-45

单击上图中的 Yes 按钮，如图 4-46 所示。

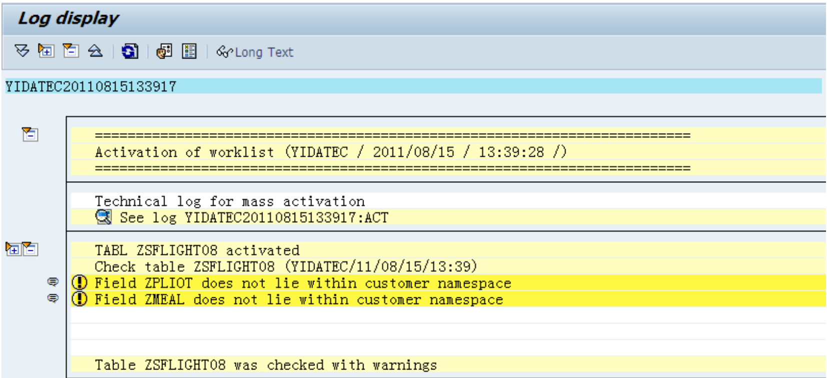


图 4-46

步骤四：测试新建表增强，使用事务 SE16，单击应用工具栏中的按钮，显示结果如图 4-47 所示。

Data Browser: Table SFLIGHT08: Selection Screen

Number of Entries

CARRID		to	
CONNID		to	
FLDATE		to	
PRICE		to	
CURRENCY		to	
PLANETYPE		to	
SEATSMAX		to	
SEATSOCC		to	
PAYMENTSUM		to	
SEATSMAX_B		to	
SEATSOCC_B		to	
SEATSMAX_F		to	
SEATSOCC_F		to	
ZPLIOT		to	
ZMEAL		to	

图 4-47

4.3.5　函数增强

函数增强即代码增强，是指利用代码完成某项功能，一般和其他增强共同完成某项业务需要。

实例：事务 BC425_08 是用来预定航班的，想实现在预定“AA”航班时给出提示信息。

步骤一：SAP 增强项目管理器（第一画面）。

在项目处输入要建立的工程名，单击画面中的 创建 按钮，如图 4-48 所示。

Project Management of SAP Enhancements

Project ZFIELD01 Create

Subobjects
- Attributes
- Enhancement Assignment
- Components
- Documentation

Display Change

图 4-48

编辑工程描述，如图 4-49 所示。

Attributes of Enhancement Project ZFIELD01

Enhancement assignments Components

Project ZFIELD01
Short text 函数增强测试

Administration data
Package
Original language EN
Created by YIDATEC 2011/08/15
Last changed on/by

Activation
Project Status
Changed

图 4-49

步骤二：SAP 增强项目管理器（第一画面：保存工程）。

单击标准工具栏中的“保存”按钮，如图 4-50 所示。

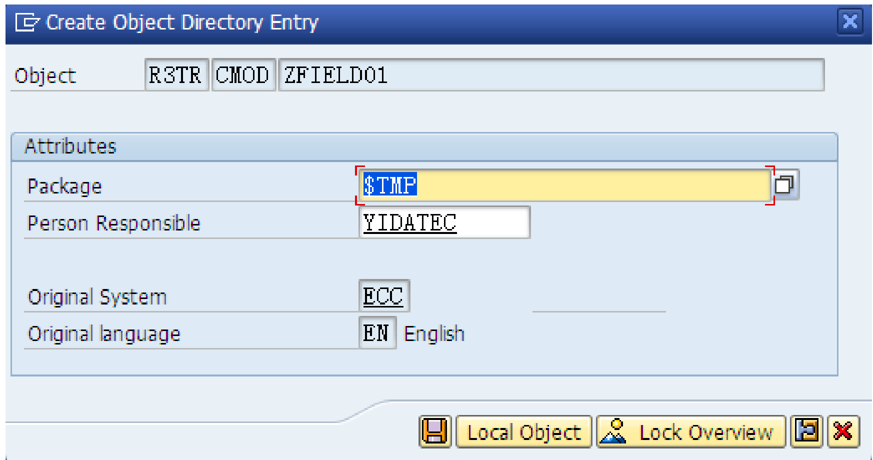


图 4-50

单击上图中的“保存”按钮，效果如图 4-51 所示。



图 4-51

步骤三：单击Enhancement assignments按钮，编辑增强点，如图 4-52 所示。

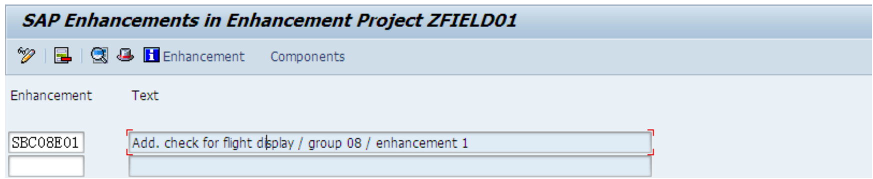


图 4-52

步骤四：保存组件。

单击标准工具栏中的“保存”按钮，如图 4-53 所示。



图 4-53

步骤五：选中增强点，单击Components按钮，编辑增强组件，如图 4-54 所示

Display ZFIELD01

Enhancement assignments　Enhancement

Project				ZFIELD01 字段增强测试
Enhancement	Impl		Exp	SBC08E01 Add. check for flight display / group 08 / enhancement 1
Function exit				EXIT_SAPBC425_FLIGHT08_001

图 4-54

步骤六：双击EXIT_SAPBC425_FLIGHT08_001 组件，编辑增强代码，如图 4–55 所示。

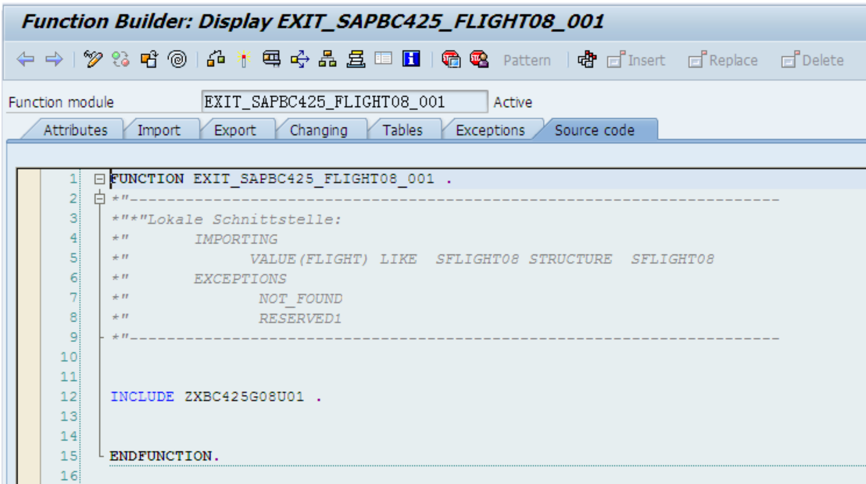


图 4–55

双击文件ZXBC425G08U01 . ，提示警告信息，如图 4–56 所示。

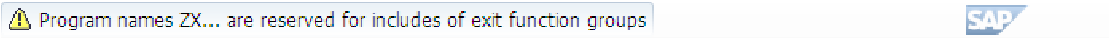


图 4–56

按〈Enter〉键弹出如下提示窗口，如图 4–57 所示



图 4–57

单击 Yes 按钮，如图 4–58 所示。

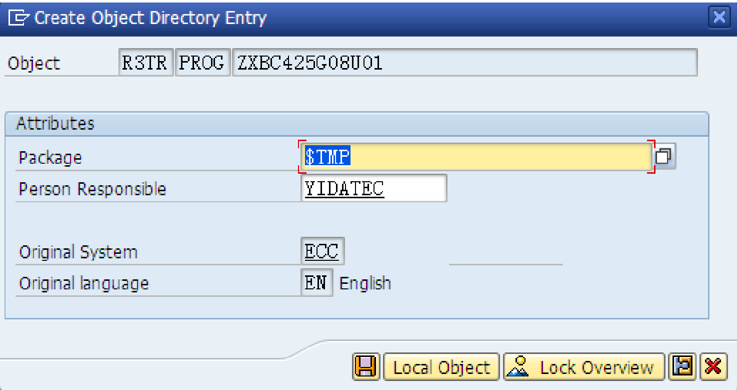


图 4–58

单击上图中的“保存”按钮，编辑 Include 文件，如图 4-59 所示。

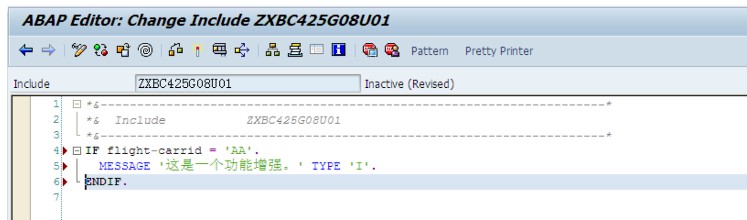


图 4-59

步骤七：保存、检查及有效化 Include 文件。

单击标准工具栏中的保存按钮，如图 4-60 所示



图 4-60

单击应用工具栏中的“检查”按钮，如图 4-61 所示。



图 4-61

单击应用工具栏中的“有效化”按钮，如图 4-62 所示。



图 4-62

步骤八：激活工程组件。

单击标准工具栏中的“后退”按钮，退回至如图 4-63 所示画面，单击应用工具栏中的“有效化”按钮，如图 4-64 所示。

Change ZFIELD01

Enhancement assignments　Enhancement

Project				ZFIELD01 函数增强测试
Enhancement	Impl		Exp	SBC08E01 Add. check for flight display / group 08 / enhancement 1
Function exit	✔			EXIT_SAPBC425_FLIGHT08_001

图 4-63



图 4-64

步骤九：激活工程。

单击标准工具栏中的“后退”按钮，退回至第一画面，单击应用工具栏中的“有效化”按钮，如图 4-65 所示。



图 4-65

步骤十：测试增强结果，运行事务代码 BC425_08，输入参数并按〈Enter〉键，结果如图 4-66 所示



图 4-66

4.3.6 菜单增强

菜单增强是指利用标准菜单中留下的接口新增相应的其他功能，一般和代码增强共同完成某项业务需要。

实例：事务 BC425_08 是用来预定航班的，想实现此画面来查询相关历史信息。

步骤一：SAP 增强项目管理器（第一画面）。

在项目处输入要建立的工程名，单击画面上的 创建 按钮，如图 4-67 所示。

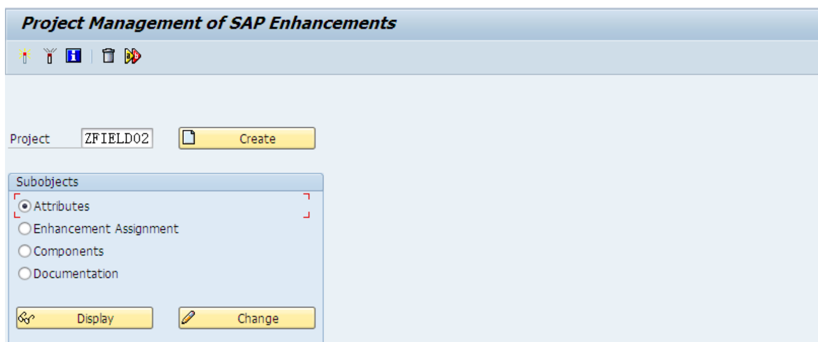


图 4-67

编辑工程描述，如图 4-68 所示。

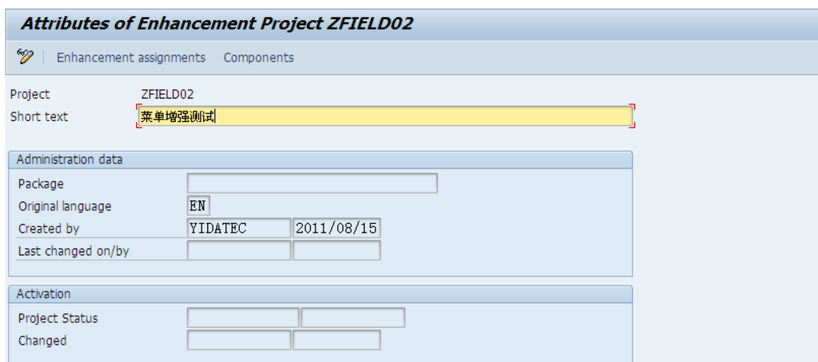


图 4-68

步骤二：SAP 增强项目管理器（第二画面：保存工程）。

单击标准工具栏中的“保存”按钮，如图 4-69 所示。

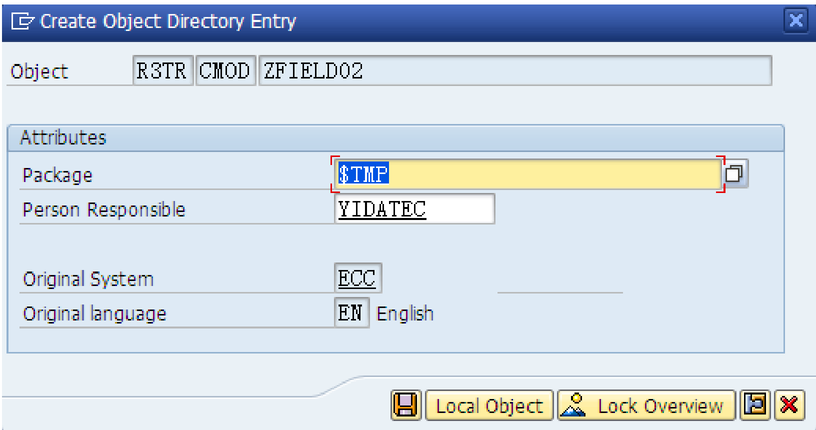


图 4-69

单击上图中的“保存”按钮。

步骤三：单击Enhancement按钮，如图 4-70 所示。编辑增强点并单击标准工具栏中的“保存”按钮保存组件，如图 4-71 所示。

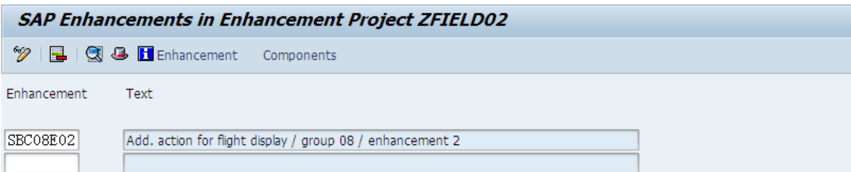


图 4-70



图 4-71

步骤四：选中增强点，单击 Components 按钮，编辑增强组件，单击按钮切换到变更状态，如图 4-72 所示。

Change ZFIELD02

Enhancement assignments　Enhancement

Project				ZFIELD02 菜单增强测试
Enhancement	Impl		Exp	SBC08E02 Add. action for flight display / group 08 / enhancement 2
Function exit	✔			EXIT_SAPBC425_FLIGHT08_002
Menu exit	✔			SAPBC425_FLIGHT08　+EXT

图 4-72

双击+EXT，编辑菜单如图 4-73 所示。

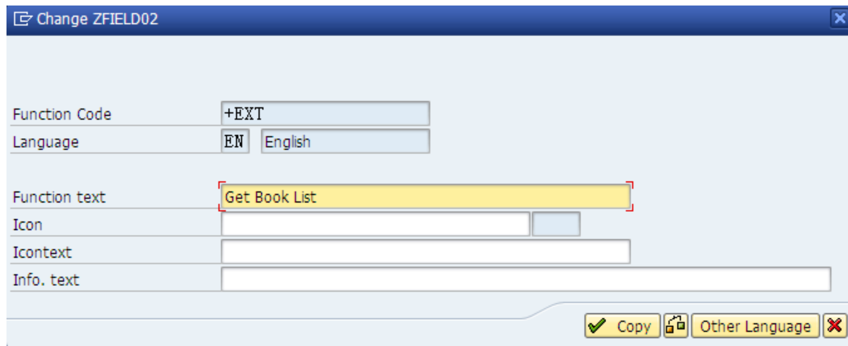


图 4-73

注：此处的编辑方式与菜单编辑器中的编辑方式一致。
Function text：功能菜单项的文本描述。
Icon：功能菜单项的图标。
Icontext：功能菜单项的图标描述。
Info.text：功能菜单项的提示信息，用于鼠标单击前提示。

单击 Copy 按钮，如图 4-74 所示。



图 4-74

输入传输请求，单击✔按钮。
步骤五：为菜单项设置功能。
双击EXIT_SAPBC425_FLIGHT08_002，编辑菜单赋予的功能代码，如图 4-75 所示。

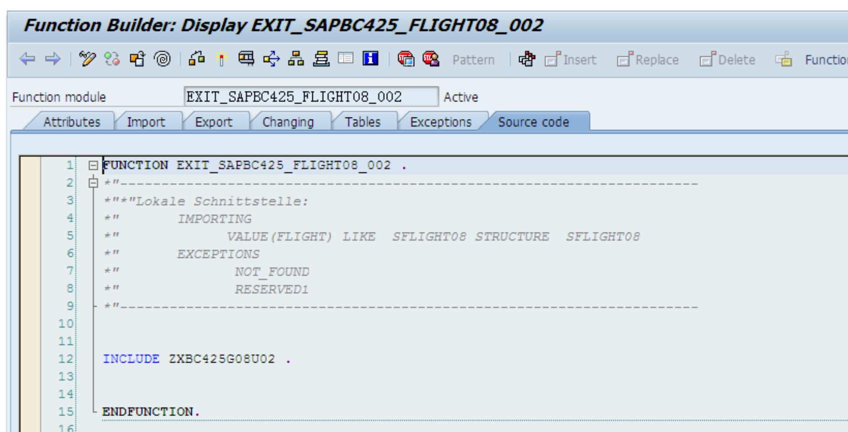


图 4-75

双击 Include 程序ZXBC425G08U02 .，提示警告信息如图 4–76 所示。

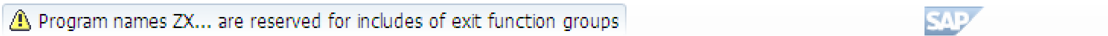


图 4–76

按〈Enter〉键弹出如下提示窗口，如图 4–77 所示。

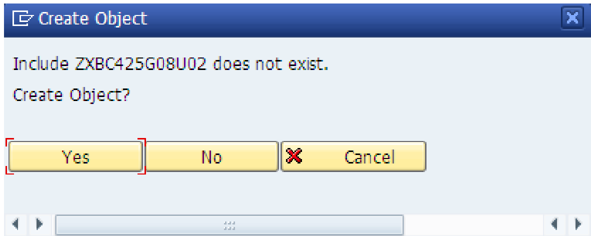


图 4–77

单击 Yes 按钮，结果如图 4–78 所示。

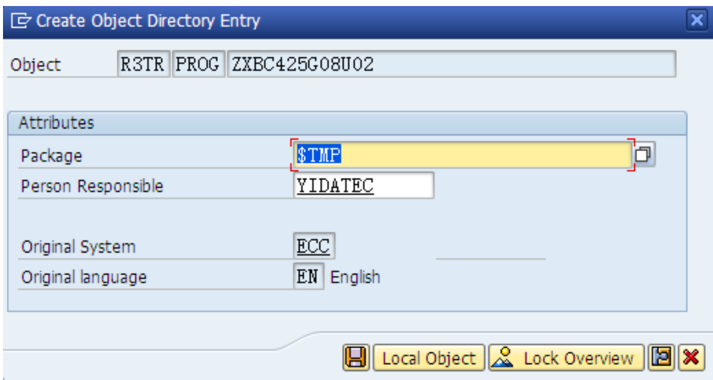


图 4–78

单击上图中的“保存“按钮，编辑 Include 文件如图 4–79 所示。

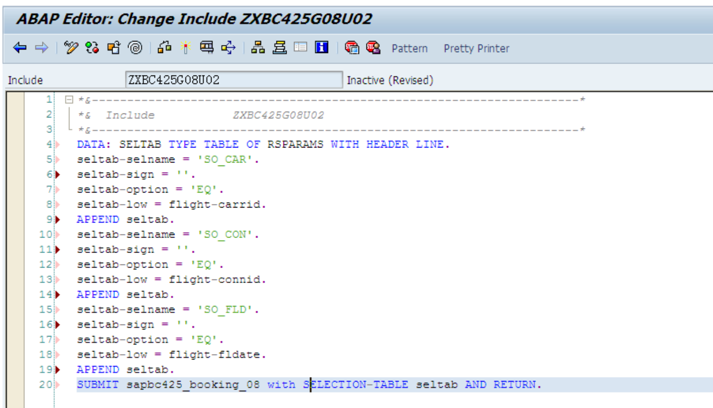


图 4–79

步骤六：保存、检查及有效化 Include 文件。

单击标准工具栏中的“保存“按钮，如图 4-80 所示。



图 4-80

单击应用工具栏中的“检查“按钮，如图 4-81 所示。



图 4-81

单击应用工具栏中的“有效化“按钮，如图 4-82 所示。



图 4-82

步骤七：激活工程组件。

单击标准工具栏中的“后退”按钮，退回至如图 4-83 所示画面，单击应用工具栏中的“有效化”按钮，如图 4-84 所示。

Change ZFIELD02

Enhancement assignments Enhancement

Project				ZFIELD02 菜单增强测试
Enhancement	Impl		Exp	SBC08E02 Add. action for flight display / group 08 / enhancement 2
Function exit	✔			EXIT_SAPBC425_FLIGHT08_002
Menu exit	✔			SAPBC425_FLIGHT08 +EXT

图 4-83



图 4-84

步骤八：激活工程。

单击标准工具栏中的“后退”按钮，退回至第一画面，单击应用工具栏中的“有效化”按钮，如图 4-85 所示。



图 4-85

步骤九：测试增强结果，运行事务代码 BC425_08，选中以下增强菜单路径，如图 4-86 所示。

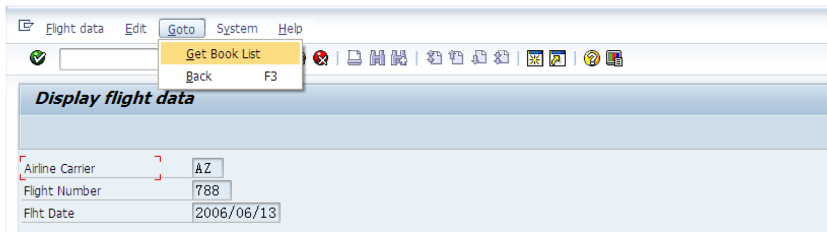


图 4-86

显示结果如图 4-87 所示。

Copy of program SAPBC425_BOOKING_00

Bookings per airline carrier

Booking number	Booking date	Customer number
AA 0017		
2006/06/14		
00000137	2006/03/06	00004484
00000136	2006/01/21	00002031
00000138	2006/05/23	00001142
00000139	2006/05/06	00002626
00000135	2006/03/09	00001428
00000134	2006/01/23	00001464
00000133	2006/01/16	00004410
00000132	2006/05/18	00002616

图 4-87

4.3.7 屏幕增强

屏幕增强是指利用标准屏幕中留下的字屏幕新增相应的其他项目，一般和代码增强共同完成某项业务需要。

实例：事务 BC425_08 是用来预定航班的，想实现此画面中更多的信息。

步骤一：SAP 增强项目管理器（第一画面）。

在“Project”文本框中输入要建立的工程名，单击画面上的 创建 按钮，如图 4-88 所示。

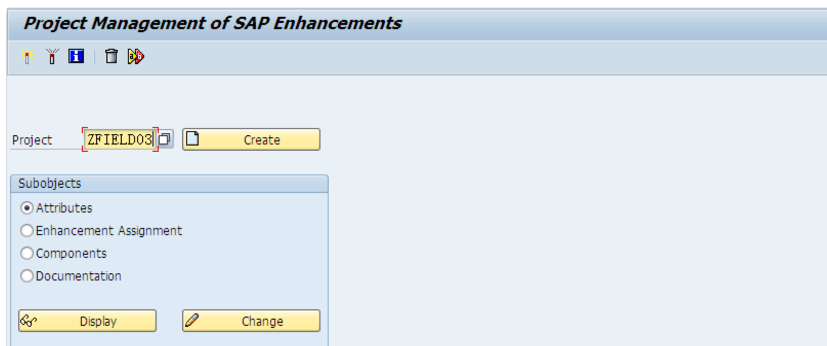


图 4-88

编辑工程描述，如图 4-89 所示。

图 4-89

步骤二：SAP 增强项目管理器（第二画面：保存工程）。

单击标准工具栏中的“保存”按钮，如图 4-90 所示。

图 4-90

单击上图中的“保存”按钮，如图 4-91 所示。



图 4-91

步骤三：单击Enhancement按钮，如图 4-92 所示。编辑增强点并单击标准工具栏中的“保存”按钮保存组件，如图 4-93 所示。

图 4-92



图 4-93

步骤四：选中增强点，单击 Components 按钮，编辑增强组件，单击按钮切换到变更状态，如图 4-94 所示。

Display ZFIELD03

Enhancement assignments　Enhancement

Project		△		ZFIELD03 屏幕增强测试
Enhancement	Impl	△	Exp	SBC08E03 Add. data for flight display / group 08 / enhancement 3
Function exit		■		EXIT_SAPBC425_FLIGHT08_003
Screen exit				SAPBC425_FLIGHT08　0200　SUB　SAPLXBC425G08　0500

图 4-94

双击 SAPLXBC425G08　0500，编辑屏幕如图 4-95 所示。

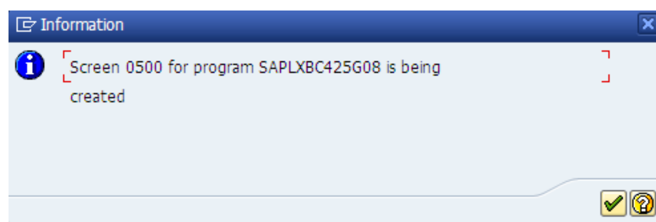


图 4-95

单击上图中的按钮，编辑属性页签如图 4-96 所示。

Screen Painter: Change Screen for SAPLXBC425G08

Layout

Screen number 500 Active(Revised)

Attributes　Element list　Flow logic

Short description	测试子屏幕		
Original language	DE German	Package	$TMP
Last changed on/by	2011/08/16 07:38:44		
Last generation	2011/08/16 07:38:45		

Screen type: ○ Normal　◉ Subscreen　○ Modal dialog box　○ Selection screen

Settings: ☐ Switch off runtime compress　☐ Template - non-executable　☐ Hold Scroll Position　☐ Without Application Toolbar

Other attributes

Next Screen	500	
Cursor position		
Screen group		
Lines/Columns Occupied	2	45
Mainten.	27	120
Context menu FORM ON CTMENU		

图 4-96

单击 Layout 按钮，编辑如图 4-97 所示。

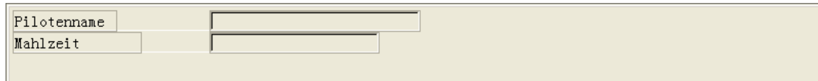


图 4–97

屏幕元素生成如图 4–98 所示。

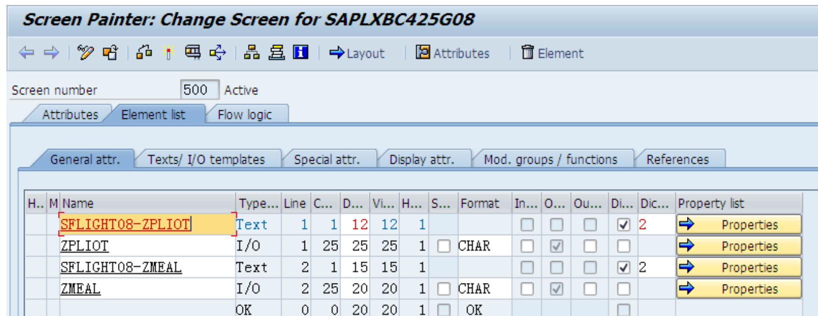


图 4–98

单击标准工具栏中的“保存”按钮，如图 4–99 所示。

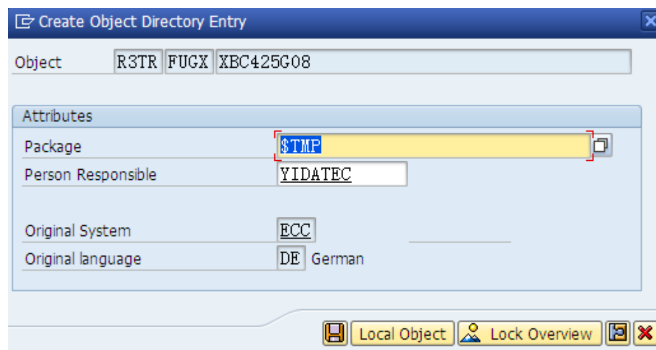


图 4–99

单击上图中的“保存”按钮，如图 4–100 所示。



图 4–100

双击屏幕字段ZPLIOT，弹出提示窗口，如图 4–101 所示。

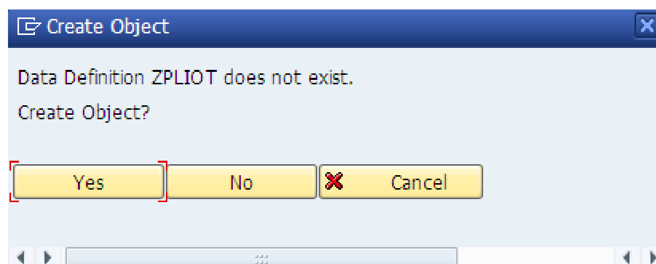


图 4–101

单击 Yes 按钮，如图 4–102 所示。

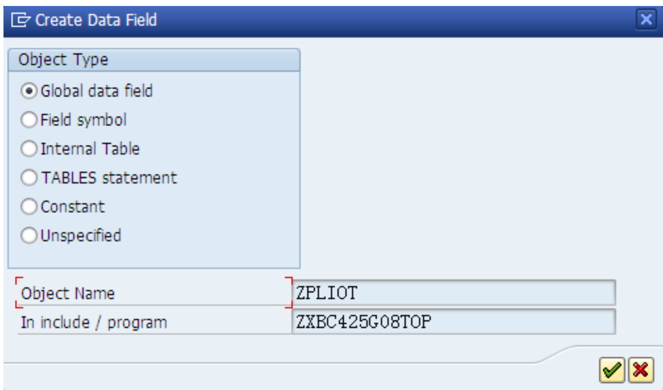


图 4–102

按〈Enter〉键或单击上图中的按钮，如图 4–103 所示。

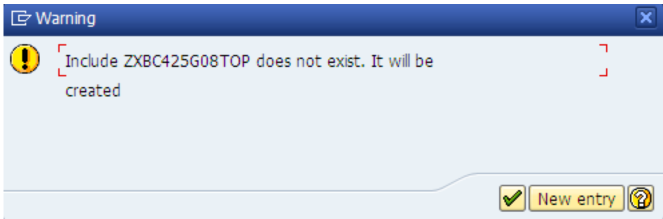


图 4–103

按〈Enter〉键或单击上图中的按钮，创建 Include 文件，如图 4–104 所示

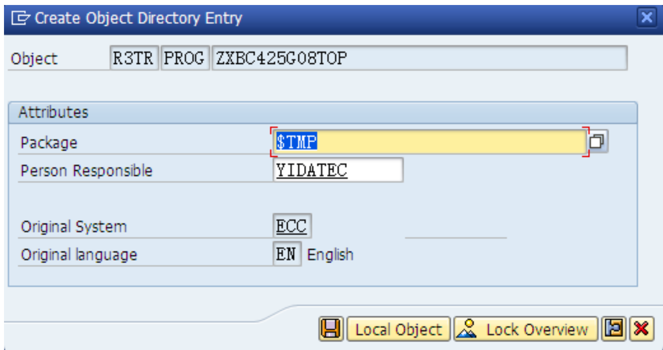


图 4–104

单击上图中的“保存”按钮，如图 4–105 所示。



图 4–105

编辑 Include 文件，如图 4–106 所示，并单击标准工具栏中的“保存”按钮保存文件，如图 4–107 所示。

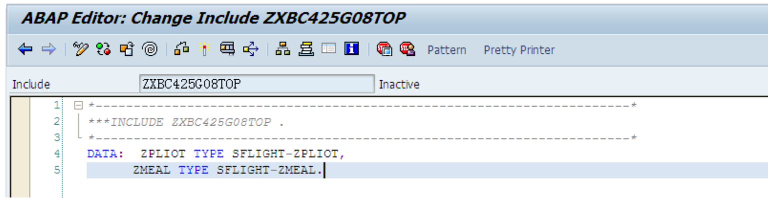


图 4-106



图 4-107

单击应用工具栏中的“检查”按钮，如图 4-108 所示。



图 4-108

单击应用工具栏中的“有效化”按钮，如图 4-109 所示。



图 4-109

单击标准工具栏中的“后退”按钮，退回至屏幕编辑画面，单击应用工具栏中的“检查”按钮，如图 4-110 所示。



图 4-110

单击应用工具栏中的“有效化”按钮，如图 4-111 所示。



图 4-111

步骤五：为屏幕赋值设置功能。

双击EXIT_SAPBC425_FLIGHT08_003 ，编辑屏幕赋值的功能代码，如图 4-112 所示。

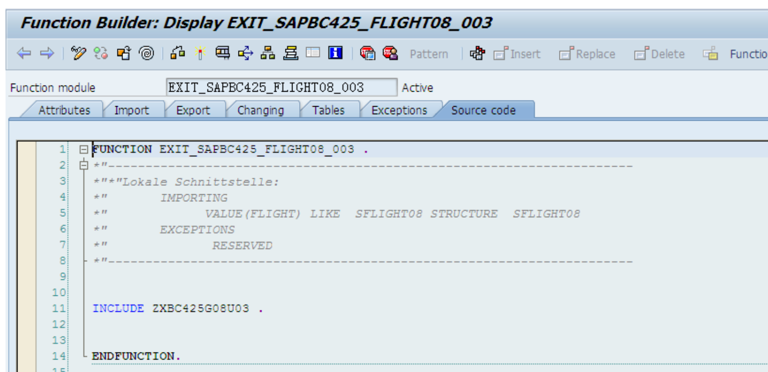


图 4-112

双击 Include 程序ZXBC425G08U03 .，提示警告信息，如图 4–113 所示。

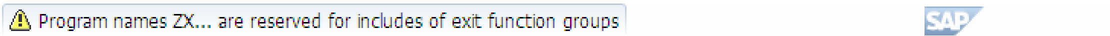


图 4–113

按〈Enter〉键弹出提示窗口，如图 4–114 所示。



图 4–114

单击 Yes 按钮，如图 4–115 所示。

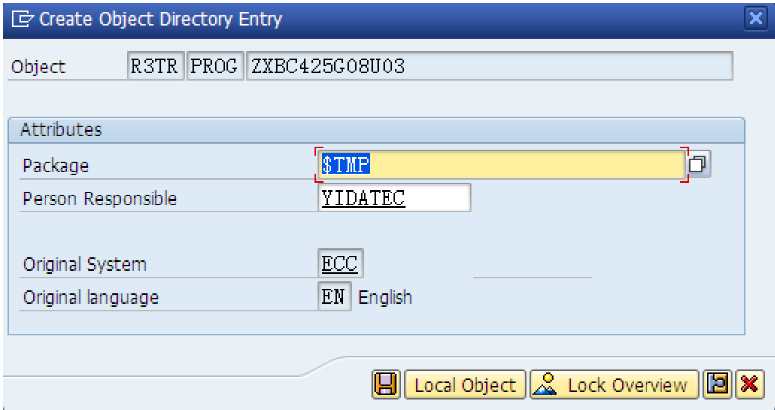


图 4–115

单击上图中的“保存”按钮，编辑 Include 文件，如图 4–116 所示。

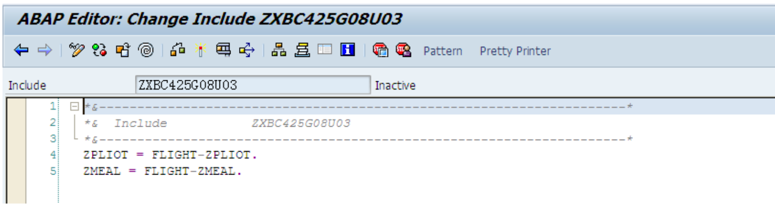


图 4–116

步骤六：保存、检查及有效化 Include 文件。

单击标准工具栏中的“保存”按钮，如图 4–117 所示。



图 4–117

单击应用工具栏中的“检查”按钮，如图 4−118 所示。



图 4−118

单击应用工具栏中的“有效化”按钮，如图 4−119 所示。



图 4−119

步骤七：激活工程组件。

单击标准工具栏中的“后退”按钮，退回至如图 4−120 所示画面，单击应用工具栏中的“有效化”按钮，如图 4−121 所示。

Change ZFIELD03

Enhancement assignments Enhancement

Project				ZFIELD03 屏幕增强测试
Enhancement	Impl		Exp	SBC08E03 Add. data for flight display / group 08 / enhancement 3
Function exit	✔			EXIT_SAPBC425_FLIGHT08_003
Screen exit	✔			SAPBC425_FLIGHT08 0200 SUB SAPLXBC425G08 0500

图 4−120



图 4−121

步骤八：激活工程。

单击标准工具栏中的“后退”按钮，退回至第一画面，单击应用工具栏中的“有效化”按钮，如图 4−122 所示。



图 4−122

步骤九：测试增强结果，运行事务代码 BC425_08，输入查询条件并按〈Enter〉键，如图 4−123 所示。

Display flight data

Airline Carrier	AZ
Flight Number	788
Flht Date	2006/06/13

Flight information

Airfare	12.29
Airline Currency	USD
Plane Type	146-200
Max. capacity	10
Occupied econ.	10
Pilot name	Mark
Meal	1

图 4−123

4.4 第三代增强的应用

BADI（Business Add-Ins）是 SAP 在 R/3 release 4.6A 后增加的增强技术。总的来说，BAdIs 克服了之前出口的所有缺点，并且用最新的面向对象技术实现，可谓是万千宠爱于一身。不过在项目中，用 BAdIs 的顾问还是比较少，多数是用 Customer Exit 和 User Exit 完成。

其实，BADI 就是综合了 SAP 之前一些增强技术的优点，在标准程序预留了出口，通过面向对象的接口技术，使增强更加灵活。一个出口可以有多个实现，但是一个 client 只有一个是起作用的。通过实验发现它是按创建的先后顺序及是否 active 来判断的。不过一般一个实现就够了，只能说 SAP 的技术架构确实是比较先进。

下面来举个例子说明一下。

4.4.1 BADI增强

实例：事务 BC425_CHNG08 是用来更改预定航班的，想用 BADI 增强实现此画面中的更多信息。

步骤一：按照以下路径新建程序，如图 4-124 所示。

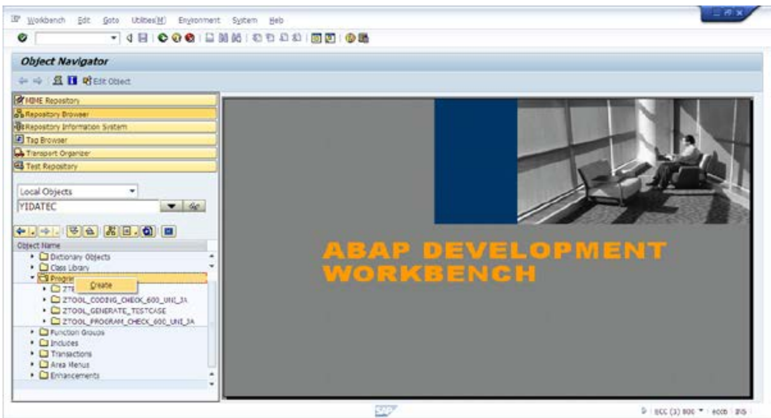


图 4-124

注：此处增强一般有两种方式。

- 用“Module pool”类型的程序。
- 用函数组作为增强的程序。

输入文件名并附 Top 选项如图 4-125 所示。

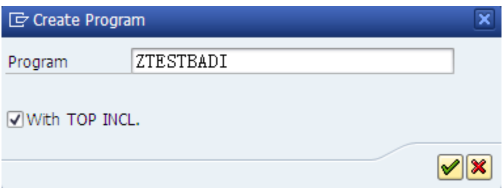


图 4-125

按〈Enter〉键或单击上图中的按钮，创建 Top Include 文件如图 4–126 所示。



图 4–126

按〈Enter〉键或单击上图中的按钮，编辑程序属性如图 4–127 所示。

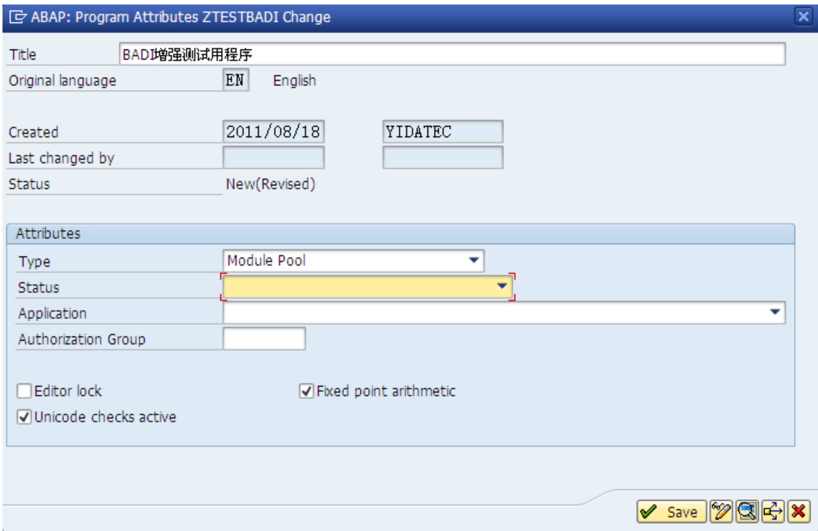


图 4–127

按〈Enter〉键或单击上图中的 Save 按钮，如图 4–128 所示。

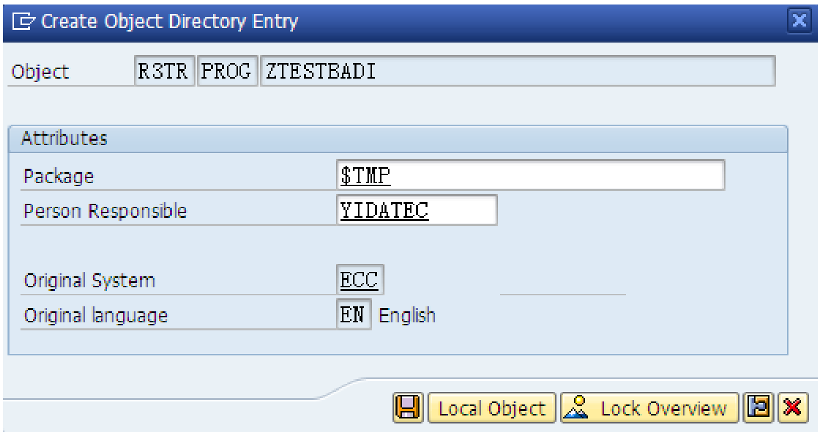


图 4–128

按〈Enter〉键或单击上图中的按钮，如图 4–129 所示。

Attributes for program ZTESTBADI saved

图 4-129

步骤二：编辑 Top Include 文件，保存、检查及有效化 Include 文件，如图 4-130 所示。

Include ZBADITOP Inactive (Revised)

```
*&---------------------------------------------------------------------*
*& Include ZBADITOP                                   Module Pool      ZTESTBADI
*&
*&---------------------------------------------------------------------*

PROGRAM  ZTESTBADI.
TABLES: SFLIGHT08.
DATA:  ZPLIOT LIKE SFLIGHT08-ZPLIOT,"Name of pilot
       ZMEAL LIKE SFLIGHT08-ZMEAL."Main meal of a flight
DATA: R_VAR TYPE REF TO IF_EX_BC425_08FLIGHT2."Business Add-In Interface IF_EX_BC425_99FLI
```

图 4-130

单击标准工具栏中的保存按钮，如图 4-131 所示。

Program ZTESTBADI saved

图 4-131

单击应用工具栏中的检查按钮，如图 4-132 所示。

Program ZTESTBADI is syntactically correct

图 4-132

单击应用工具栏中的有效化按钮，如图 4-133 所示。

Object(s) activated

图 4-133

步骤三：创建屏幕、编辑、保存。

按照以下路径创建屏幕，如图 4-134 所示。

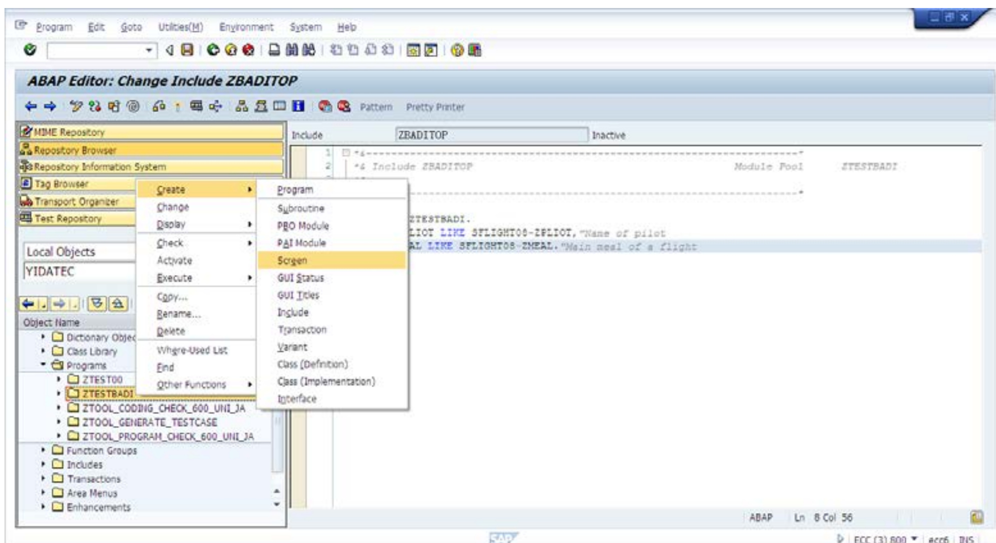


图 4-134

输入屏编号如图 4-135 所示。



图 4-135

按〈Enter〉键或单击上图中的✔按钮，编辑屏幕属性如图 4-136 所示。

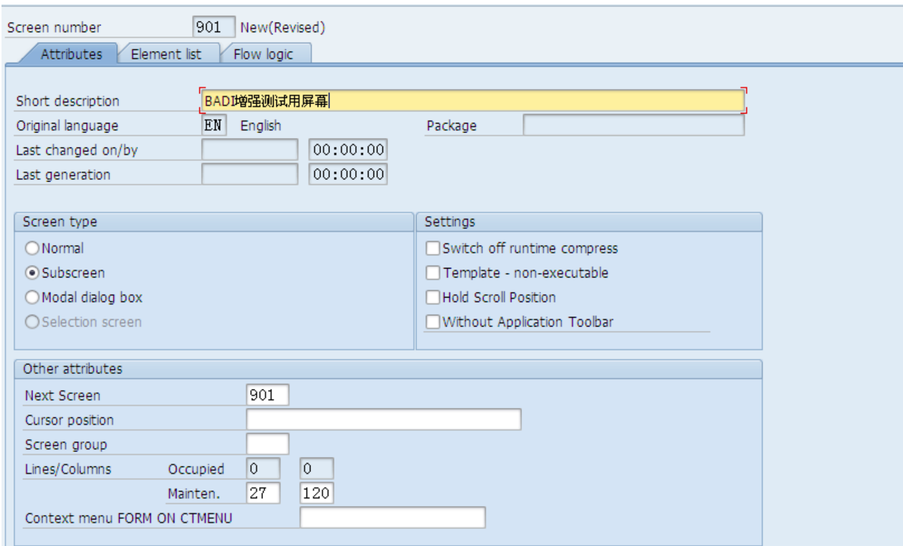


图 4-136

编辑屏幕布局如图 4-137 所示。



图 4-137

生成屏幕元素如图 4-138 所示。

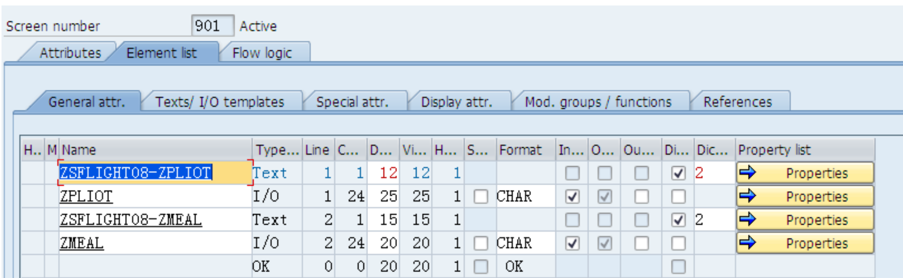


图 4-138

编辑屏幕控制逻辑如图 4-139 所示。

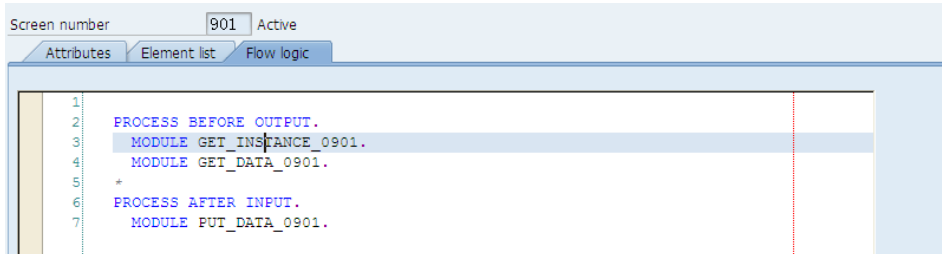


图 4-139

单击标准工具栏中的保存按钮，保存编辑屏幕，如图 4-140 所示。



图 4-140

主程序生成如下：

```
*&---------------------------------------------------------------------*
*& Module Pool         ZTESTBADI
*&
*&---------------------------------------------------------------------*
*&
*&
*&---------------------------------------------------------------------*


INCLUDE ZBADITOP                    .    " global Data

INCLUDE ZBADIO01                       .  " PBO-Modules

INCLUDE ZBADII01                        .  " PAI-Modules
```

Top 文件源码编辑如下：

```
*&---------------------------------------------------------------------*
*& Include ZBADITOP                    Module Pool      ZTESTBADI
*&
*&---------------------------------------------------------------------*

PROGRAM   ZTESTBADI.
TABLES:   SFLIGHT08.
DATA:     ZPLIOT LIKE SFLIGHT08-ZPLIOT,"Name of pilot
          ZMEAL LIKE SFLIGHT08-ZMEAL."Main meal of a flight
DATA:   R_VAR TYPE REF TO IF_EX_BC425_08FLIGHT2."Business Add-In Interface IF_EX_BC425
_99FLIGHT2
```

PBO 模块 GET_INSTANCE_0901 编辑如下：

```
*---------------------------------------------------------------------*
***INCLUDE ZBADIO01 .
*---------------------------------------------------------------------*
*&--------------------------------------------------------------------*
*&      Module  GET_INSTANCE_0901  OUTPUT
*&--------------------------------------------------------------------*
*       IF Inintial
*---------------------------------------------------------------------*
MODULE GET_INSTANCE_0901 OUTPUT.
  CALL METHOD CL_EXITHANDLER=>GET_INSTANCE_FOR_SUBSCREENS
    CHANGING
      INSTANCE                         = R_VAR
    EXCEPTIONS
      NO_REFERENCE                     = 1
      NO_INTERFACE_REFERENCE           = 2
      NO_EXIT_INTERFACE                = 3
      DATA_INCONS_IN_EXIT_MANAGEM      = 4
      CLASS_NOT_IMPLEMENT_INTERFACE    = 5
      OTHERS                           = 6.
  IF SY-SUBRC <> 0.
    MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
               WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ENDIF.

ENDMODULE.                 " GET_INSTANCE_0901  OUTPUT
```

PBO 模块 GET_DATA_0901 编辑如下：

```
*&--------------------------------------------------------------------*
*&      Module  GET_DATA_0901  OUTPUT
*&--------------------------------------------------------------------*
*       Get Data From PGM
*---------------------------------------------------------------------*
MODULE GET_DATA_0901 OUTPUT.

  CALL METHOD R_VAR->GET_DATA
    IMPORTING
      E_FLIGHT = SFLIGHT08.
  ZPLIOT = SFLIGHT08-ZPLIOT.
  ZMEAL = SFLIGHT08-ZMEAL.
ENDMODULE.                 " GET_DATA_0901  OUTPUT
```

PAI 模块 PUT_DATA_0901 编辑如下：

```
*---------------------------------------------------------------------*
```

```
***INCLUDE ZBADII01 .
*----------------------------------------------------------------------*
*&---------------------------------------------------------------------*
*&      Module  PUT_DATA_0901  INPUT
*&---------------------------------------------------------------------*
*       Put Data to PGM
*----------------------------------------------------------------------*
MODULE PUT_DATA_0901 INPUT.

  SFLIGHT08-ZPLIOT = ZPLIOT.
  SFLIGHT08-ZMEAL = ZMEAL.
  CALL METHOD R_VAR->PUT_DATA
    EXPORTING
      I_FLIGHT = SFLIGHT08.

ENDMODULE.                 " PUT_DATA_0901  INPUT
```

步骤四：检查并有效化程序。

按照以下路径检查程序，如图 4-141 所示，其效果如图 4-142 所示。

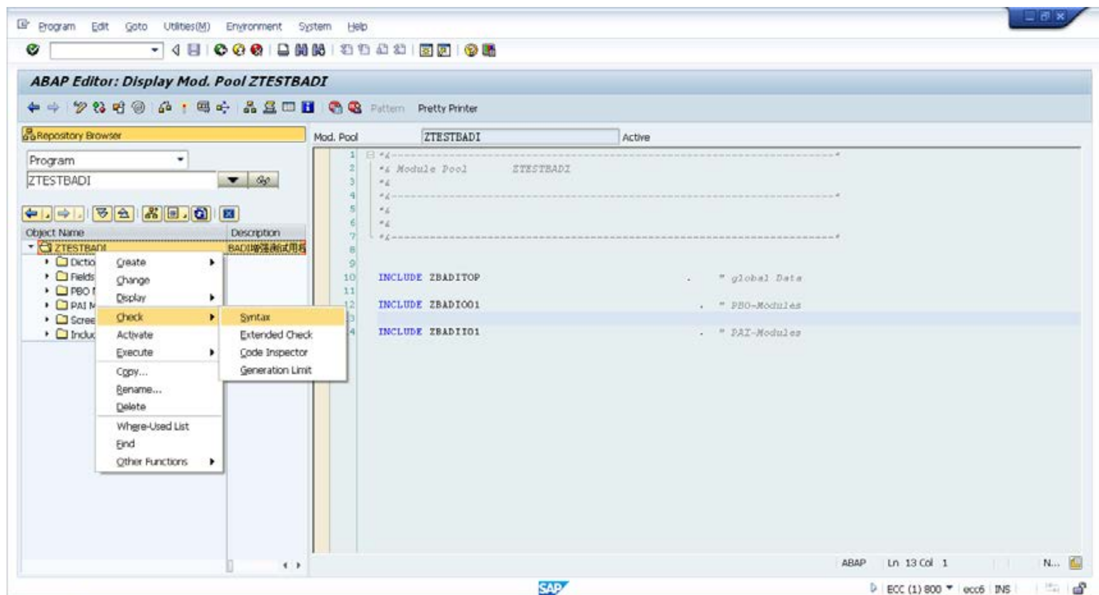


图 4-141



图 4-142

按照以下路径激活程序，如图 4-143 所示，其效果如图 4-144 所示。

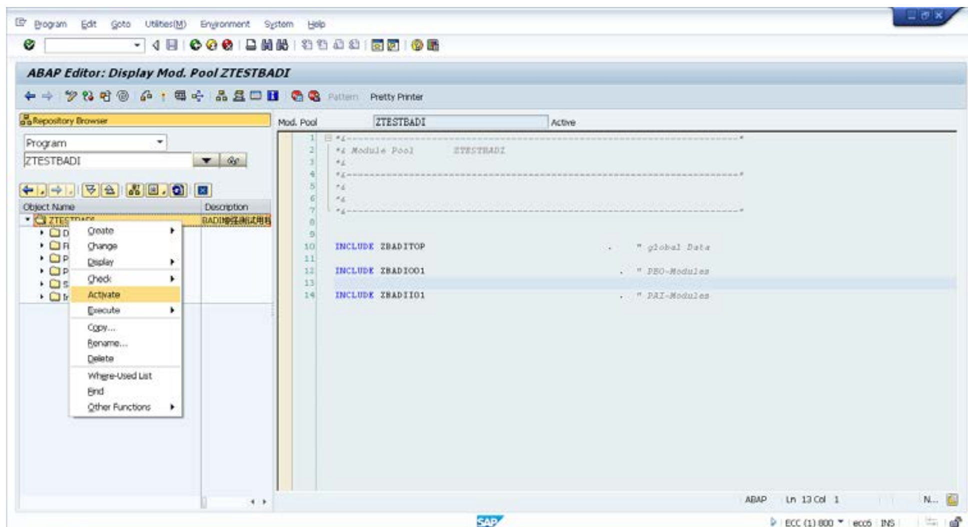


图 4-143



图 4-144

步骤五：运行增强实装事务（事务代码：SE19），创建增强实例。

填写要增强的 BADI，单击 Create Impl. 按钮，如图 4-145 所示。

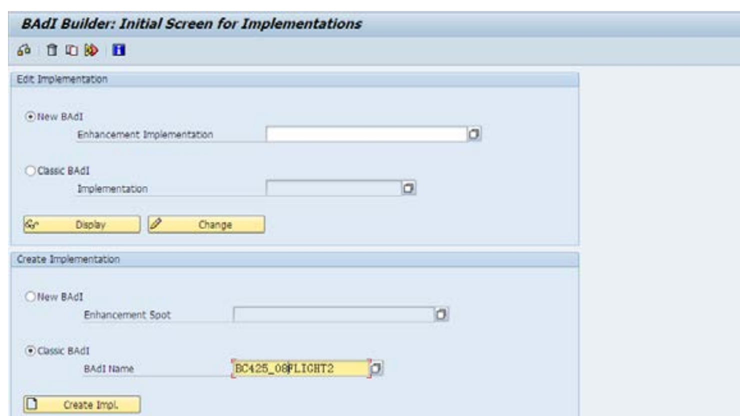


图 4-145

填写实装实例名称如图 4-146 所示。

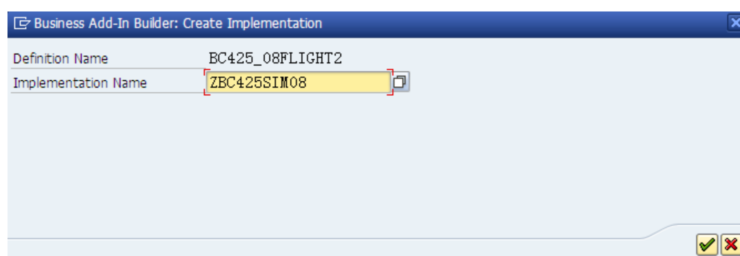


图 4-146

按〈Enter〉键或单击上图中的按钮，编辑实例描述，如图 4–147 所示。

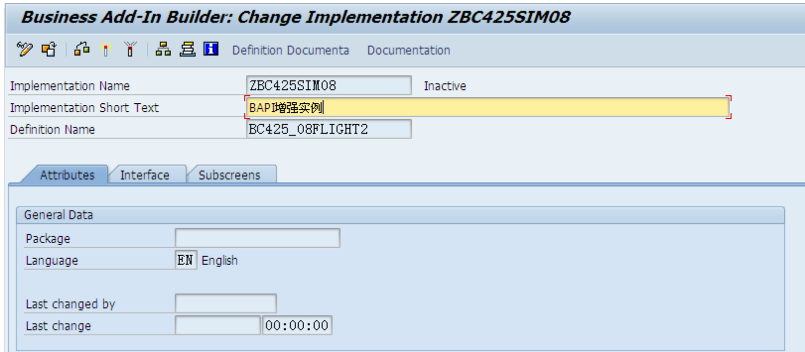


图 4–147

单击标准工具栏中的保存按钮，保存编辑实例，如图 4–148 所示。

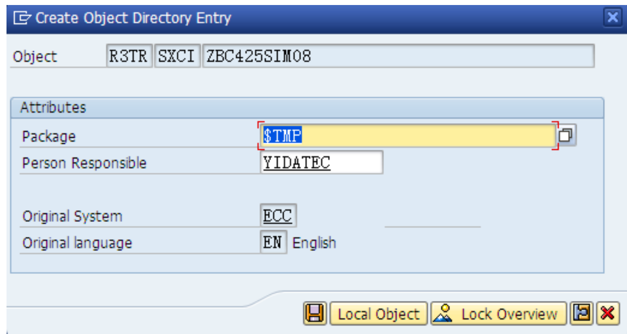


图 4–148

按〈Enter〉键或单击上图中的按钮，如图 4–149 所示。

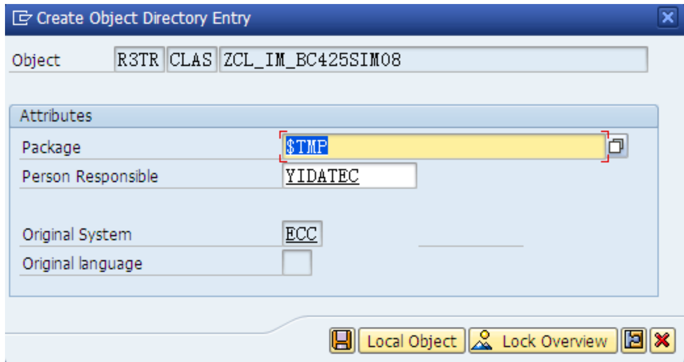


图 4–149

按〈Enter〉键或单击上图中的按钮，如图 4–150 所示。

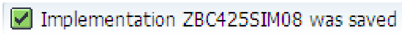




图 4–150

步骤六：编辑增强实例。

选中 Interface 标签页，如图 4–151 所示。

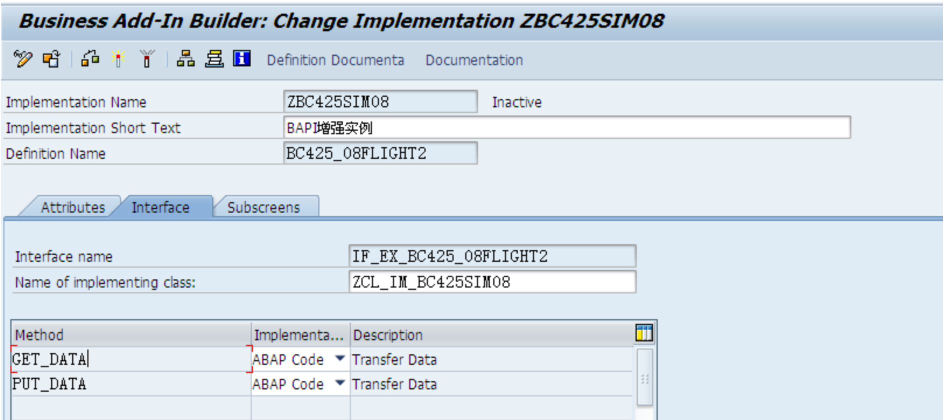


图 4–151

双击GET_DATA方法，编辑如图 4–152 所示。

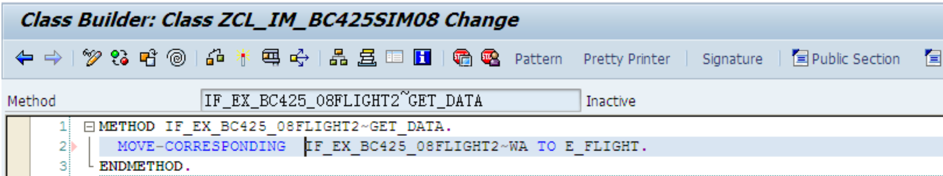


图 4–152

单击标准工具栏中的保存按钮，保存实例方法，如图 4–153 所示。



图 4–153

单击应用工具栏中的检查按钮，如图 4–154 所示。



图 4–154

单击应用工具栏中的有效化按钮，如图 4–155 所示。



图 4–155

单击应用工具栏中的回退按钮至实例编辑页面，双击PUT_DATA 方法，编辑如图 4–156 所示，并对方法保存、检查及有效化。

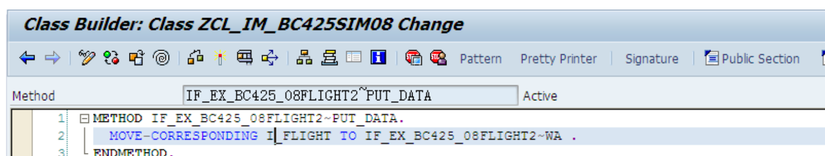


图 4-156

选中 Subscreens 标签页，编辑子屏幕属性如图 4-157 所示。

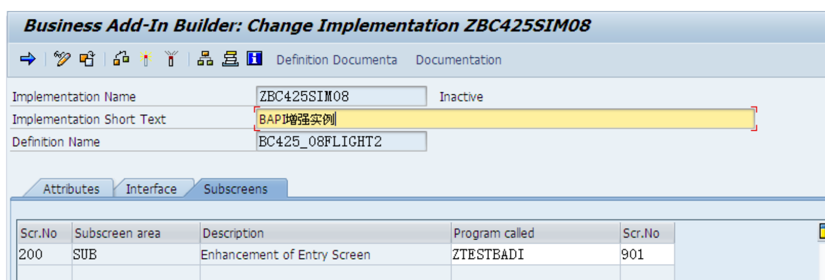


图 4-157

步骤七：保存、检查及有效化增强实例。

单击标准工具栏中的保存按钮，如图 4-158 所示。



图 4-158

单击应用工具栏中的检查按钮，如图 4-159 所示。

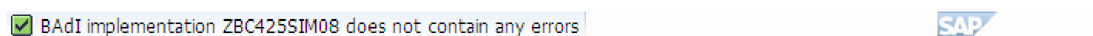


图 4-159

单击应用工具栏中的有效化按钮，如图 4-160 所示。



图 4-160

步骤八：测试增强结果，运行事务代码 BC425_CHNG08，输入查询条件并按回车键，如图 4-161 所示。

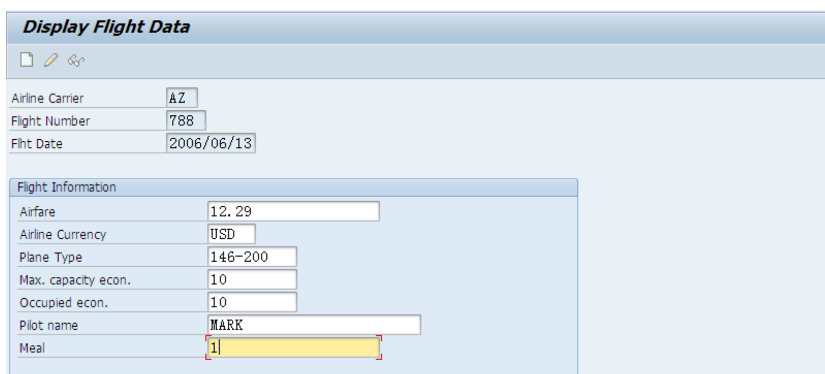


图 4-161

4.4.2 BTE增强

BTE（Business Transaction Events）是 SAP 的一种增强方式，技术上讲是介于 Customer Exits 和 BADI 的产物。

它主要有两种类型的接口：

1．P/S 函数模块：

■ 一个事件触发时，允许开始一个或多个附加操作，它们不影响标准的 R/3 程序。

■ 多个操作相互不影响。

■ 附加开发的组件只能从 SAP 导入数据。

■ 可能使用附加的检查，例如权限、查重等。

2．处理函数模块：

■ 会发生双向的数据交换，因此限制了一个接口可以附加开发的组件个数。

■ 事件触发时，一个标准处理流程只能被一个外部流程所替换。

■ 对于一个 process interface，如果你使用 SAP 合作伙伴附加开发的组件，运行时会使用它的增强；如果后来对此 process interface 使用自己开发的组件，在运行时将会用此增强替换合作伙伴的。

BTE 增强比较核心的理念是使一个接口重复利用，完成多种实现并且互不干扰。而技术实现的时候，就采用在标准程序中 Call 固定的一个 function，然后再用这个 function 动态调用自己定义的增强 function，从而达到预期目的。

这样的接口，在标准程序中常常是以 OPEN_FI_PERFORM_或者 OUTBOUND_CALL_开头的函数。由于采用这种技术架构，SAP 就使这些接口分为 Made by SAP、Made by SAP's Partner 和 Made by SAP's Customer，三方可以各取所需。

至于 Publish and Subscribe Interface 和 Process Interface 的区别，主要还是体现在用途上。前者是给外部软件提供信息的，而后者主要用于 R/3 流程。不过前者也可以生成或修改一些 additional data。

BTE 主要用于总账，应收、应付和销售分销模块。

实例：事务 FB01 是用来创建凭证的，想用 BTE 增强实现凭证各项被分配给某个固定字符串。

步骤一：按照以下路径 IMG Menu Path：Financial Accounting→Financial Accounting Global Settings→Business Transaction Events 或运行事务：FIBF，如图 4–162 所示。

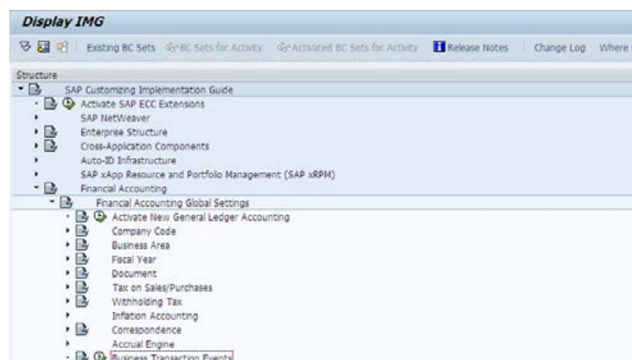


图 4–162

按照以下路径查询 BTE，如图 4-163 所示。

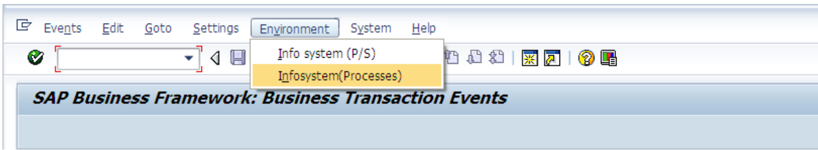


图 4-163

输入查询条件，单击按钮，如图 4-164 所示。

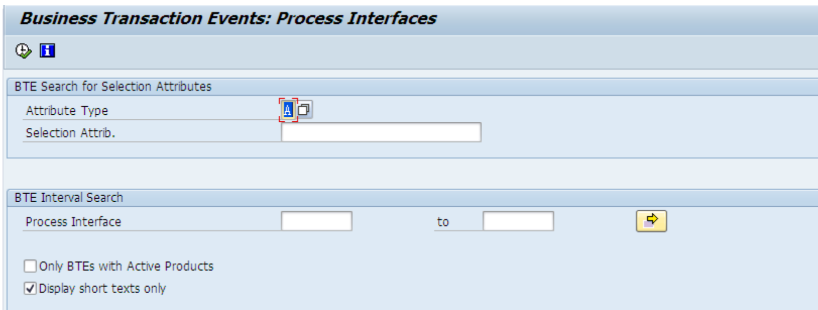


图 4-164

选中 BTE 过程，单击 Sample function module 按钮，如图 4-165 所示。

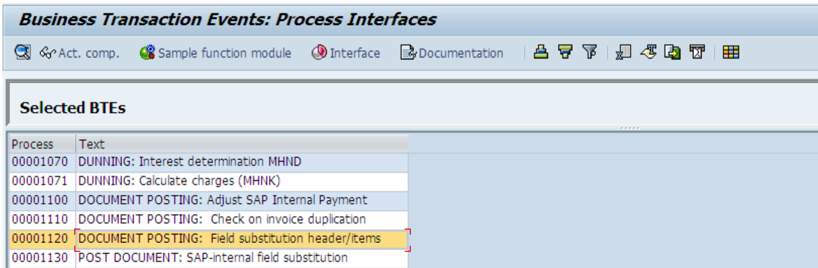


图 4-165

单击应用工具栏中的拷贝按钮，如图 4-166 所示。

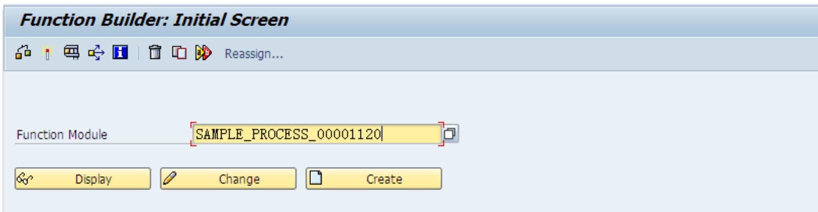


图 4-166

弹出如图 4-167 所示画面。

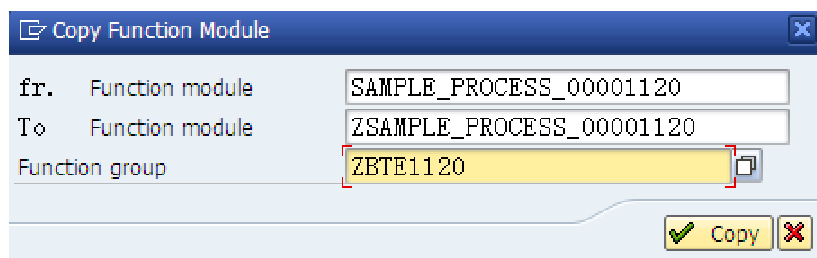


图 4-167

单击上图中的Copy按钮，如图 4-168 所示。



图 4-168

注：执行此步需先创建函数组，如图 4-169 所示。



图 4-169

编辑新拷贝的函数，单击Change按钮，如图 4-170 所示。

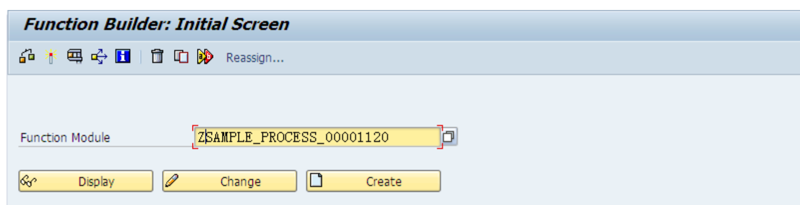


图 4-170

编辑代码如图 4-171 所示，并将其保存、检查至有效化。

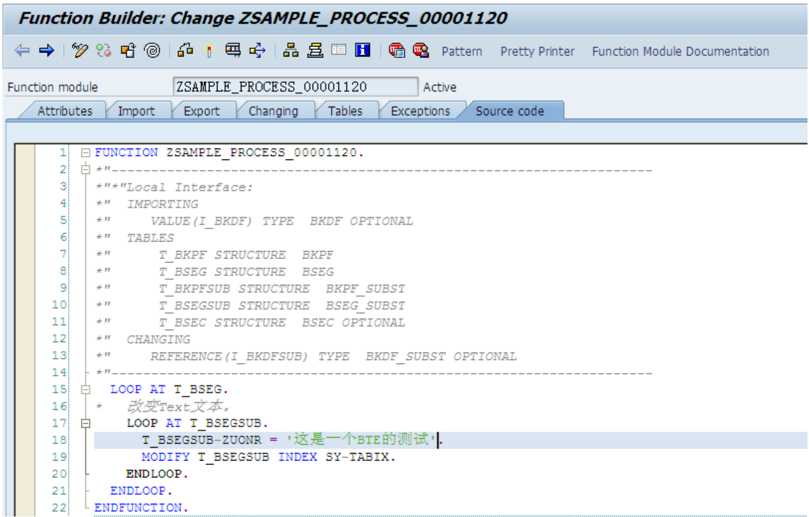


图 4-171

步骤二：单击按钮返回至 BTE 初始画面，按照以下路径定制 BTE 产品，如图 4-172 所示。

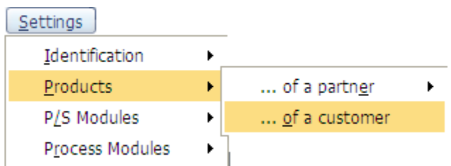
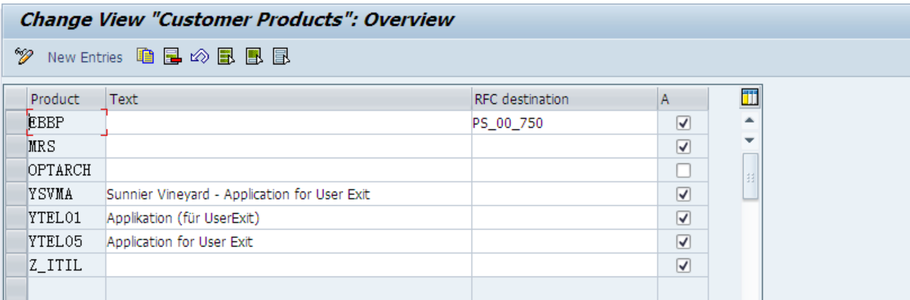


图 4-172

单击应用工具栏中的New Entries按钮，如图 4-173 所示。

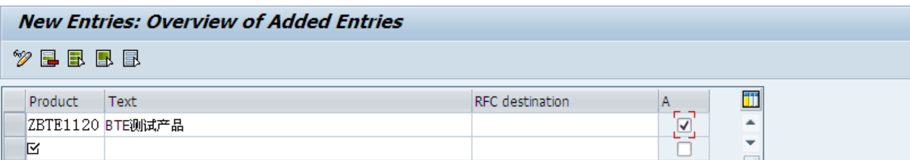

Change View "Customer Products": Overview

New Entries

Product	Text	RFC destination	A
EBBP		PS_00_750	☑
MRS			☑
OPTARCH			☐
YSVMA	Sunnier Vineyard - Application for User Exit		☑
YTEL01	Applikation (für UserExit)		☑
YTEL05	Application for User Exit		☑
Z_ITIL			☑

图 4-173

编辑定制 BTE 产品信息如图 4-174 所示。


New Entries: Overview of Added Entries

Product	Text	RFC destination	A
ZBTE1120	BTE测试产品		☑
			☐

图 4-174

单击标准工具栏中的保存按钮，如图 4–175 所示。

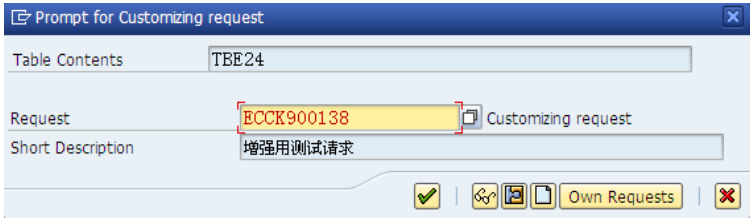


图 4–175

按〈Enter〉键或单击上图中的按钮，如图 4–176 所示。



图 4–176

步骤三：单击按钮返回至 BTE 初始画面，按照以下路径定制 BTE 过程，如图 4–177 所示。

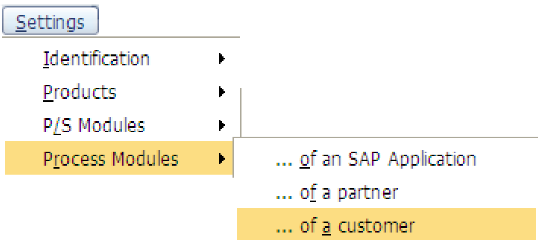


图 4–177

单击应用工具栏中的New Entries按钮，如图 4–178 所示。

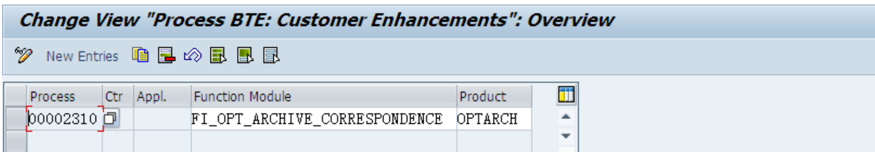


图 4–178

编辑定制 BTE 过程信息如图 4–179 所示。

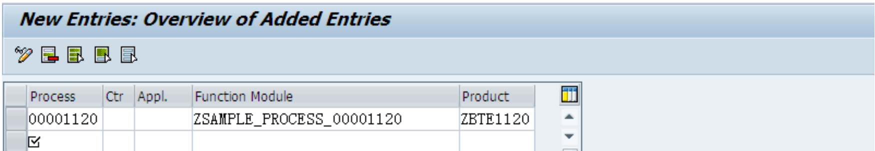


图 4–179

单击标准工具栏中的保存按钮，如图 4–180 所示。

Prompt for Customizing request

Table Contents TPS34

Request ECCK900138 Customizing request

Short Description 增强用测试请求

Own Requests

图 4-180

输入传输信息，按回车键或单击上图中的✔按钮，如图 4-181 所示。

Data was saved

图 4-181

步骤四：运行事务代码 FB01，测试增强结果。

如图 4-182 所示，填入公司代码。

Enter Company Code

Company Code 1000

图 4-182

编辑总账凭证如图 4-183 所示，其效果如图 4-184 所示。

Enter G/L Account Document: Company Code 1000

Tree on　Company Code　Hold　Simulate　Park　Editing options

Basic data　Details

Document Date 2011/08/17　Currency EUR

Posting Date 2011/08/17

Reference

Doc.Header Text

Cross-CC no.

Company Code 1000 IDES AG Frankfurt

Amount Information

Total deb. 12.00 EUR

Total cred. 12.00 EUR

2 Items (No entry variant selected)

St...	G/L acct	Short Text	D/C	Amount in doc.curr.	Loc.curr.amount	T..	Tax jurisdictn code	W	Assignment no.
✔	100000	Petty cash	Cred...	12.00	12.00				
✔	112700	HSBC (domes...	Debit	12.00	12.00				
					0.00				

图 4-183

Document 100000025 was posted in company code 1000

图 4-184

运行事务代码 FB03，显示凭证行明细如图 4-185 所示。

图 4-185

相关事务代码如表 4-1 所示。

表 4-1

事 务 代 码	描　　述
FIBF	Maintenance transaction BTE
BERE	Business Event Repository
BERP	Business Processes
BF31	Application modules per Event
BF32	Partner Modules per Event
BF34	Customer Modules per Event
BF41	Application Modules per Process
BF42	Partner Modules per Process
BF44	Customer Modules per Process

4.5　第四代增强（Code Enhancement）的应用

这是一个普通的 report，与一般程序没有任何区别，如图 4-186 所示。

```
*&---------------------------------------------------------------------*
*& Report  ZTEST
*&
*&---------------------------------------------------------------------*
*&
*&
*&---------------------------------------------------------------------*

REPORT  ZTEST.
WRITE 'My First PGM'.
```

图 4-186

步骤一：光标放置在要增强的代码位置，插入一个增强点（enhancement-point），如图 4-187 所示。



图 4-187

输入名称，如图 4-188 所示。

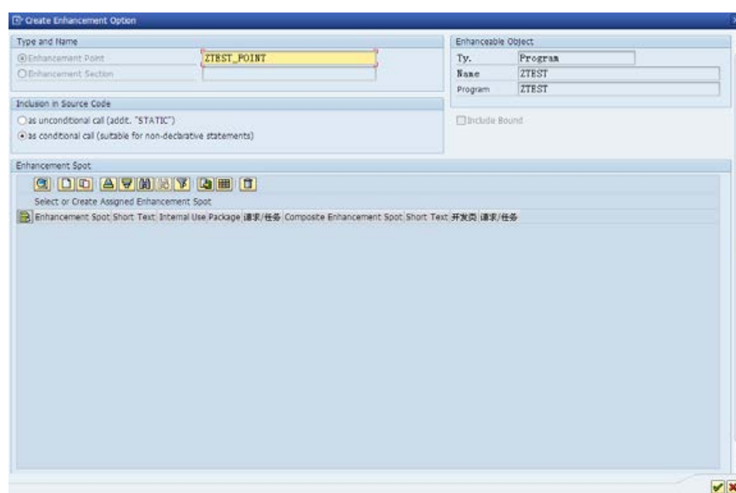


图 4-188

单击按钮，创建增强点如图 4-189 所示。

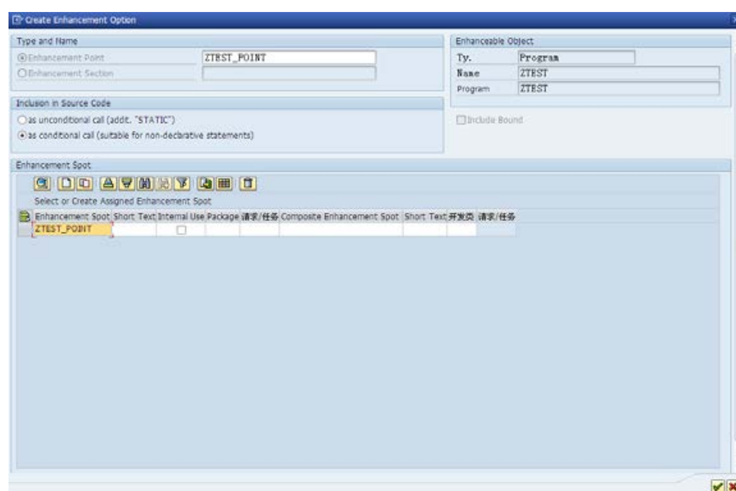


图 4-189

单击上图中的按钮，如图 4–190 所示。

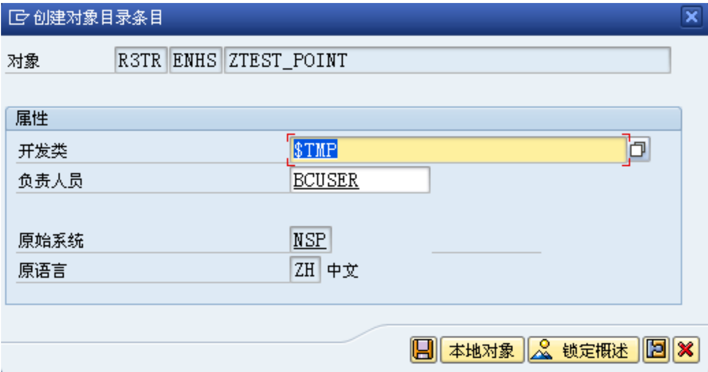


图 4–190

单击上图中的保存按钮，增强点插入如图 4–191 所示。

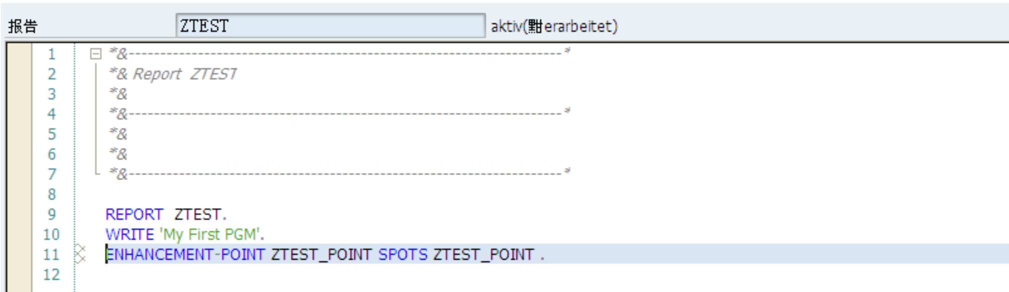


图 4–191

单击标准工具栏中的保存按钮，如图 4–192 所示。



图 4–192

注：实际项目中要做的事情是实现（implementation）SAP 预留的 enhancement point，具体操作如下。

步骤二：单击应用工具栏中的按钮，切换到增强状态，如图 4–193 所示。



图 4–193

按照下列路径创建增强，如图 4−194 所示。



图 4−194

填写增强实例及其所存包名称，如图 4−195 所示。

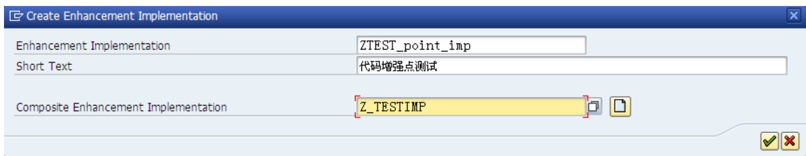


图 4−195

单击上图中的按钮，如图 4−196 所示。

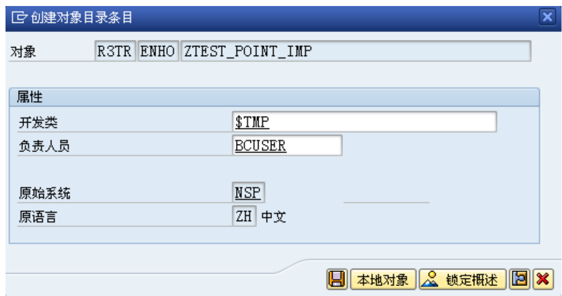


图 4−196

单击上图中的保存按钮，增强之后的代码如图 4−197 所示。



图 4−197

单击应用工具栏中的 Enhancements 按钮，编译增强结果，如图 4–198 所示。

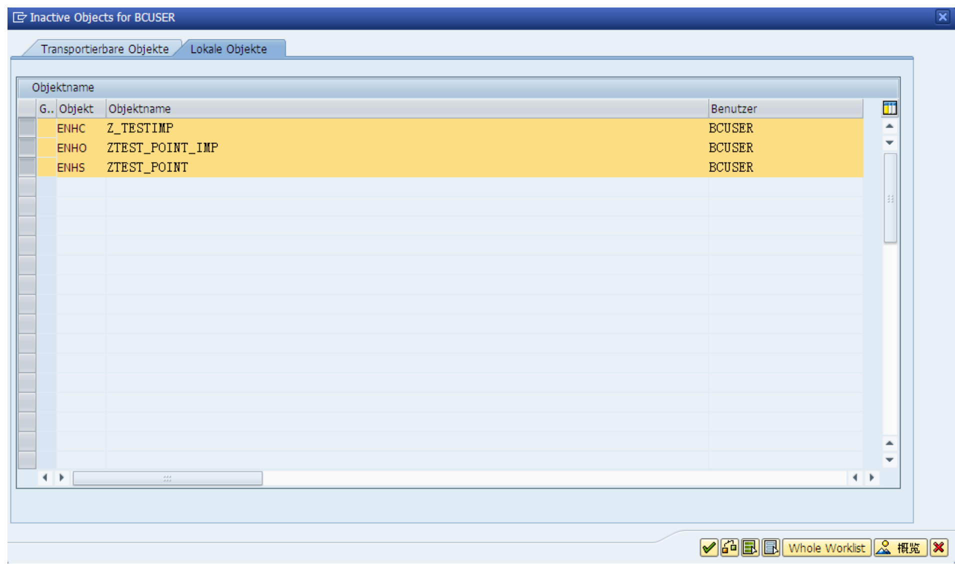


图 4–198

选择增强对象，单击上图中的按钮，如图 4–199 所示。



图 4–199

步骤三：单击应用工具栏中的 Active <-> Inactive 按钮，切换到程序编辑环境，单击应用工具栏中的按钮，编译整个程序。

步骤四：单击应用工具栏中的按钮，测试增强结果如图 4–200 所示。



图 4–200

对于 ENHANCEMENT-SECTION，定义和实现的方法与 ENHANCEMENT-POINT 一样。两者的区别是：

- ENHANCEMENT-POINT 没有代码，只有一个预留点，允许在这个位置插入新代码（implementation）。
- ENHANCEMENT-SECTION 和 END-ENHANCEMENT-SECTION 之间有代码，implementation 之后，替换旧代码，只执行新代码，原来的代码不再执行。

SAP 还提供 Modification 机能（相关事务代码：SE95 SPAU SPAM）。建议不要轻易使用，一般在系统升级时会被覆盖变更后的代码。

第 5 章　ABAP面试问题及答案

5.1　语法

5.1.1　基础知识

1．CHECK、EXIT、RETURN 命令的区别？

■ CHECK

- CHECK 后面要跟一个表达式，当表达式值为假（false）时，CHECK 发生作用，退出循环（LOOP）或处理程序（Processing Block）。
- 如果 CHECK 出现在循环中，则发生作用时，退出的是当前一次循环操作，程序会继续执行下一次循环操作，其作用类似于 Continue（Java 或 C++中 Continue 也是如此）。
- 如果 CHECK 出现在循环以外，则发生作用时，退出的是当前执行的程序块，例如一个 FORM、METHOD 或 EVENT。

■ EXIT

- EXIT 如果出现在循环中，退出的是整个循环操作，程序会从循环结束处开始继续执行，其作用相当于 Java 与 C++中的 Break。
- EXIT 如果出现在循环之外，退出的是当前执行的程序块（Processing Block），例如一个 FORM、METHOD 或 EVENT，其作用与 RETURN 类似。

■ RETURN

- RETURN 用来退出当前执行的程序块，例如一个 FORM、METHOD 或 EVENT。不管是否出现在循环（LOOP）中，RETURN 都会退出当前执行的程序块，而不仅仅是退出循环。
- 虽然 ABAP 中 EXIT 和 RETURN 都可以用来实现退出当前执行的语句块，但 SAP 的帮助文件建议只在循环中使用 EXIT，其他情况下要退出当前执行进程，使用 RETURN。

2．初始化内表有几种方式？

■ 要初始化有或没有表头的内表，用 REFRESH 语句。

■ 如果使用没有表格工作区域的内表，可以使用 CLEAR 语句代替 REFRESH 语。

■ 如果使用有表头行的内表，CLEAR 语句仅清除表格工作区域，要重置整个内表而不清除表格工作区域，使用 REFRESH 语句或 CLEAR 语句。

■ 使用 REFRESH 或 CLEAR 初始化内表后，系统保持在内存中保留的空间。可以用 FREE 语句释放内存。

3．一个程序如何调用另外一个程序？

调用其他可执行程序使用“SUBMIT+程序名”语句，使用 WITH 关键字，可传递选择画面参数。使用 AND RETURN 关键字，可在该程序执行完毕后，自动返回调用它的程序。

4．在一个程序中如何调用其他事物代码？

调用其他事务代码使用“CALL TRANSACTION + 事务代码”，使用 AND SKIP FIRST SCREEN 可以跳过第一个画面，进入下一个画面。

5．在进行画面跳转时，CALL SCREEN 与 LEAVE TO SCREEN 的区别？

CALL SCREEN 是将正在运行的画面挂起，进入所调用的画面，当使用 LEAVE TO SCREEN 0 时，能够返回原画面，可理解为嵌套调用；而 LEAVE TO SCREEN 是立即结束本画面的执行，调用所指定的画面，在调用画面中，无法再返回原画面。

6．与索引相关的系统字段有哪些？分别有什么作用？

SY-INDEX 存储 DO 循环的索引，SY-TABIX 存储 LOOP 循环到内表的第几行，SY-DBCNT 存储从数据库中读取到的满足条件的记录条数，SY-LISEL 存储被选中行的内容，在事件 AT LINE-SELECTION 时起作用，该系统字段中的内容到下一个屏幕的时候还存在。

7．LOOP 循环中 AT 控制逻辑有哪些？分别有什么作用？

- AT FIRST 功能：循环中第一条记录时调用。
- AT LAST 功能：循环中最后一条记录时调用。
- AT NEW 功能：循环中当指定字段之前（包括指定字段）的组合和上一条记录不一样时调用。
- AT END OF 功能：循环中当指定字段之前（包括指定字段）的组合和下一条记录不一样时调用。
- AT 事件的注意点：AT 事件中的工作区不同于 LOOP 循环的工作区，该工作区中位于指定字段后面的值将不会被读取，如果需要用到则需 read 重新抽取一次。

8．MESSAGE 消息有哪些类型和含义？如何自定义 MESSAGE 消息类？

A：Abend，立刻终止当前事务，弹出终止对话框。

X：Exit，立刻终止当前事务，进入 DUMP 的界面。

E：错误，画面的状态栏会弹出一个错误提示信息。

I：信息，弹出一个消息对话框。

S：确认，画面的状态栏会弹出一个正确提示信息。

W：警告，画面的状态栏会弹出一个警告提示信息。

可以通过语句 DISPLAY LIKE mtype 取代原错误图标；自定义 MESSAGE 消息类，通过事务代码 SE91。

9．ABAP 中，如何自定义异常类？如何捕捉异常？

通过事务代码 SE24，在 Class Type 中，选择 Exception Class 即可定义，在 TRY　CATCH 语句中就能够使用。

```
捕获异常：语法和 JAVA 相似
TRY
   …
CATCH  异常类
   …
ENDTRY.
```

10．什么是事物变式？事务变式有什么用？

事务变式是一组屏幕变式，用于预定义屏幕行为和默认值。通过使用变式功能，可以将用户不需要的字段、子屏幕及全屏幕从用户视图中取消。可以给任何输入字段设置默认值，字段也可以不用带 "Ready for Input" 状态。只能为对话和报表事务创建事务变式；变式中只能包含普通屏幕、子屏幕及对话屏幕。

开发人员可以使用 GuiXT 脚本语言通过事务变式维护对屏幕进行修改。修改屏幕布局的方式有：插入按钮、值帮助（Value Helps）、移动对象、插入屏幕等。

11．RANGE 表内的字段有哪些？分别有什么含义？

RANGE 表包括 SIGN、OPTION、LOW、HIGH 四个字段。RANGE 表和 select-option 的变量差不多，sign = 'I' 或 sign = 'E' 是指 INCLUDE 和 EXCLUDE，OPTION = 'EQ' 或其他操作符，LOW 是指最少值（操作符是范围的话），HIGH 指最大值（如果操作符是‘EQ’，该字段不起作用）。

12．对于 FIELD SYMBOL 赋值将使用什么关键字？

ASSIGN。

13．PERFORM 调用子程序时，分别对 USING、TABLES、CHANGING 3 个参数有什么影响？

USING 可以传入单值、工作区，加 VALUE 参数后只能使用，不能对值做修改，PERFORM 结束后传入参数的值还是传入之前的状态。

TABLES 只能传入一个内表，程序逻辑可以对其进行修改，PERFORM 结束后内表的值是影响后的值。

CHANGING 可以传入内表、单值，程序逻辑可以对其进行修改，PERFORM 结束后传入参数的值是影响后的值。

注意，在传入工作区、单值的情况下，USING 不加 VALUE 参数与 CHANGING 结果相同。

14．如何在程序间传送数据？

可以使用 EXPORT 语句在 ABAP/4 内存中存储数据字段簇。

语句为：EXPORT <OBJECT1> <OBJECT2> ... <OBJECTN> TO MEMORY ID <ID-NAME>。

而调用程序就会检索数据。

语句为：IMPORT <OBJECT1> <OBJECT2> ... <OBJECTN> FROM MEMORY ID <ID-NAME>。

其中：ID 参数标识唯一的数据簇。

15．SAP 和 ABAP 内存的区别？

■ 读取时使用方法不同

SAP 内存使用 SET/GET parameters 方法。

ABAP 内存使用 EXPORT 和 IMPORT 方法。

■ 共享范围不同

SAP 内存可以被所有的主 Session 访问，内存数据可以在同一个 Session 中的不同程序或者不同 Session 之间共享数据。

ABAP 只能在同个 Session 的不同程序之间共享数据。

■ 作用范围不同

SAP 内存在整个终端 Session 时间内都有效。

ABAP 内存只能在一个 Session 时间内有效。

■ 使用一般原则

SAP 内存用于屏幕默认值输入。

ABAP 内存用于模块之间传替数据。

5.1.2 应用技术

1．自定义搜索帮助有几种方式？如何实现？

■ SE11 创建搜索帮助。

■ 调用函数 F4IF_INT_TABLE_VALUE_REQUEST。

在 POV 中或者 AT SELECTION-SCREEN ON VALUE-REQUEST FOR 中使用自定义的搜索帮助可以使用函数“F4IF_FIELD_VALUE_REQUEST”。

2．怎么创建编码范围（Number Range）？

首先使用事务代码 SNUM/SNRO 创建，在使用编码范围的编号时，需要首先将其锁定（函数：NUMBER_RANGE_ENQUEUE），然后取出编号（函数： NUMBER_GET_NEXT），最后解锁（函数： NUMBER_RANGE_DEQUEUE）。

3．函数中，如何给出错误消息？

有以下两种方式：

■ 在 TABLES 参数中增加返回表（Return Table），可以将执行过程中产生的错误消息返回，在函数外对返回表做处理。

■ 定义 EXCEPTION，当出现错误时，用 RASING 语句抛异常，RASING 语句会直接影响 SY-SUBRC 的值，函数外可以直接根据 SY-SUBRC 来做处理。

4．如何优化 ABAP 程序？

■ 可通过运行事务代码 SE30，运行待检查的程序，查看各操作的百分比。

■ 避免在循环语句中使用 SELECT 语句，而是通过 FOR ALL ENTRIES IN 语句抽取数据到内表中。

■ 通过运行事务代码 SLIN 进行扩展检查，根据 SAP 提供的反馈信息优化代码。

5．前导零的处理方式

```
CALL FUNCTION  ‘CONVERSION_EXIT_ALPHA_INPUT’     添加前导 0
CALL FUNCTION  ‘CONVERSION_EXIT_ALPHA_OUTPUT’   去掉前导 0
SHIFT  物料编码  LEFT DELETING LEADING '0'.       去掉前导 0
```

6．怎样从文件服务器上读取文件？怎样写文件到文件服务器上？

从服务器上打开文件时，要确定文件的操作方式，调用语句：

```
OPEN DATASET dset FOR access IN mode [position]
                                     [os_addition]
                                     [error_handling].
```

从服务器上读取文件时，access 用 output 替换，再调用语句：

```
READ DATASET dset INTO dobj [MAXIMUM LENGTH mlen]
                             [[ACTUAL] LENGTH alen].
```

写文件到服务器上 access 用 input 替换，再在调用语句：

```
TRANSFER dobj TO dset [LENGTH len]
                      [NO END OF LINE].
```

7．在使用 ABAP 内存的程序间进行数据交换时用到的两个语句是什么？

EXPORT TO MEMORY ID 用于将数据复制到 ABAP 内存，IMPORT FROM MEMORY ID 用于将数据从 ABAP 内存复制到程序中。

在 ABAP 内存间进行交换的数据必须在两个程序中都进行声明，并包含同样的数据声明。

8．SELECT SINGLE 和 SELECT UP TO 1 ROWS 的区别是什么？

SELECT SINGLE WHERE 条件要确定唯一记录；SELECT UP TO 1 ROWS 可以确定多条记录，只取第一条，一般用来进行存在性检查。

9．清空内表命令 CLEAR、REFRESH 和 FREE 的区别？

如果内表有表头行：

CLEAR ITAB 同时清空内表数据行和表头行的内容；

CLEAR ITAB[] 只清空内表本身，保留表头行的内容；

REFRESH 只清空内表本身，保留表头行的内容。

如果内表没有表头行：

CLEAR 和 REFRESH 一样，清空内表。

CLEAR ITAB.：带表头行时只清空表头行，不带表头行时清空整个内表。

CLEAR ITAB[].：只清空整个内表对象数据，不清空表头行。

REFRESH ITAB 或 REFRESH ITAB[].：只清空整个内表对象数据，不清空表头行。

FREE ITAB.或 FREE ITAB[].：只清空整个内表对象数据，不清空表头行，同时释放内存。

10．FORM 中 USING、CHANGING、USING VALUE、 CHANGING VALUE 的区别？

- USING、CHANGING 引用传递，改变值。
- USING VALUE 值传递，值不改变。
- CHANGING VALUE 值传递，当正常结束后值改变。

11．MESSAGE 参数最多几个占位符，占位符对应的消息文本长度最长？屏幕显示消息文本的长度最长？

最多占 4 个占位符，一个&为 50 个字节长，加上 MESSAGE 自身共有的长度为 240。

所有的消息都存储在系统数据库表 T100 中，该表包括四个字段：

SPRSL（Language Key）：语言代码。

ARBGB（Message class）：消息类，用于将消息按应用分类储存。

MSGNR（Message number）：三个数字的消息序列号。

TEXT（Message text）：消息文本，是显示在屏幕中的信息文字，最多可以有 72 个字符。

消息类型共六种：A（终止）、E（错误）、I（信息）、S（成功）、W（警告）、X（退出）。

12．使用 FOR ALL ENTRIES 时的注意事项有哪些？

类似外连接的时候，用一个表的某一字段为条件，从另一数据库表中取得数据。

向 LOOP 套 SELECT 的时候用 2 个表，为了提升效率用 FOR ALL ENTRIES。

■ 使用前要检查内表是否为空，否则会查询出全部数据。
■ 要尽可能多的用关键字，否则会把重复的删掉。
■ 此子句中不能使用 LIKE、BETWEEN、IN、ORDER BY。

13．删除内表重复数据前应该做什么，如果要按照给定字段如何删除？

删除前应做 SORT 排序，如果要按照给定字段进行删除则需要 COMPARING 语句指定字段。

14．哈希表和排序表的注意事项是什么？

排序表（SORTED TABLE）与标准表相同，也具有一个逻辑索引，但其按关键字升序排序后再进行存储，其访问方式与标准表相同。

哈希表（HASHED TABLE）没有索引，只能通过关键字来访问。系统用哈希算法管理表中的数据，因而其检索到一个数据行的时间与表的行数无关。

15．怎样做带按钮的选择画面？

类似 CHECKBOX 和 RADIOBUTTON，可通过 PARAMETER 进行设置。

16．求和函数（COLLECT）的注意事项？

■ 所附加的内表必须具有扁平结构（组件中不可包含内表）。
■ 工作区必须与内表的行类型兼容（不仅仅是可相互转换）。
■ 所有的非表关键字段必须是数字类型。
■ 将新行附加至内表末尾，已存在相同表关键字值的行，不附加新行，而是将数字字段的值累计到已有数据行。

17．请列举 ABAP 程序的运行方式？它们一般表现为什么类型的程序？

可以分配事务代码并通过事务代码运行，用于完成一项系统功能，可以添加到最终用户的系统定制菜单中。

可执行程序（Report）：模块池程序。

不能直接通过事务代码运行的程序：功能组、ABAP 类库、ABAP 接口库、子程序、类型组包含程序。

18．COLLECT 和 APPEND 的区别，以及 COLLECT 命令的限制是什么？

要逐行填充内表，可以使用 APPEND、COLLECT 或 INSERT 语句。COLLECT 用于添加无相同表关键字的内表行，APPEND 可以添加相同表关键字的内表行。

COLLECT 命令的限制事项：

■ 要填充没有重复条目的内表，应该使用 COLLECT。
■ COLLECT 语句无法用于带深层结构的内表（组件中不可以包含内表）。
■ 为 COLLECT 指定的工作区域必须与内表的行类型兼容（不仅仅是可互相转换）。
■ 所有的非表关键字段必须是数字类型。

19．APPEND 与 INSERT 对内表操作的区别？

语法：

```
APPEND WA TO ITAB
```

INSERT WA INTO TABLE ITAB

前者不能用于 HASHED 表，后者不能用于 SORTED 表的 NON-UNIQUE 操作。

20．排序表还能插入和删除吗？

语法是允许的，但是规范里不推荐。因为会引起索引重排而导致的性能问题。

21．AT FIRST、AT NEW、AT END OF 有什么区别？

- AT FIRST：当前循环为内表的第一行，即当第一次循环操作时，系统会执行控制语句块。
- AT LAST：当前循环为内表的最后一行。
- AT NEW f：行组的开头，行组指的是字段 f 和 f 前面（即左边）的全部字段，如果行组中的任何字段内容不同于上一行，执行语句块中的内容。
- AT END OF f：行组的结尾，如果下一行行组中的任何字段内容不同于上一行，执行语句块中的内容。

22．AT NEW、AT END OF 指定字段之后的字段将发生怎样的变化，应该注意什么？

在 AT/END AT 语句块内部，目标区域并没有用当前表格行进行填充，而是将进行下述操作：对于 AT FIRST 和 AT LAST，系统用“*”填充所有标准关键字段内容；对于 AT NEW f 和 AT END OF f，系统用“*”填充目标区域中所有出现在指定字段 f 后面（右边）的标准关键字段。对于所有非标准关键字段，将进行初始化。

5.2 数据字典相关问题

1．激活锁对象时，产生的函数名字是什么？

首先要在 ABAP 字典中创建锁对象，才能在 ABAP 程序中设锁。创建锁对象时，系统会自动生成两个函数来进行锁管理。

用于设锁的函数为 ENQUEUE_<锁对象名>，它用于在锁表（Lock Table）中生成一个锁项（Lock Entry）。若设锁不成功的话，就会在 Return 中反映出来。

用于释放锁的函数为 DEQUEUE_<锁对象名>，它用于从锁表中删除一个锁项。在 ABAP 程序中，只需使用 "CALL FUNCITION ..." 语句就可以调用它们。

这两个锁函数是在 SAP 系统的一个特殊工作进程中执行的，专门进行锁管理。它们运行在一个单独的服务器上，而该服务器专门用于维护整个 SAP 系统的主锁表（Central Lock Table）。

有两种锁类型：

- 共享锁（shared）——只读锁，该模式允许多个用户访问指定表行，但只能读访问。任何时候都不允许写访问。
- 排他锁定（exclusive，not cumulative）——该模式允许单个用户对指定表行进行读和写访问，其他用户不能访问该行。
- 扩展排他锁定（exclusive，cumulative）——该模式避免具有读写访问权限的单个用户获得对相同表行集的进一步锁定。当使用递归例程更新时，该模式很有用。

在事务 SM12 中查看锁表的情况。

2．向 SAP 表中添加字段的方法是哪两种？

这两种方法是 Append 结构（Append Structure）和自定义 Include（Customizing Include）。Append 结构是在向表尾添加字段时创建的，自定义 Include 由 SAP 开发人员指定，使用户可以创建新字段。

3．Doman、Data Element 和 Table 之间的关系，为什么要分三层？

同样字段定义一个字段可以方便使用。

域（Doman）是定义数据元素的技术属性，具体包括数据字典中的基本内置数据类型，长度和小数位，进而确定该数据元素的数值范围。如果不同的数据元素属于相同的域（Doman），它们就有相同的属性。数据字典中，每一个数据元素必须属于一个域，以确定其基本性质，定义好数据元素可以用作结构或表类型中的字段组件或者数据库表的字段，因而这些组件字段的基本属性也都由某个域决定。

数据元素（Data Elements）相当于定义在数据字典中的 ABAP 基本类型和引用类型。

表（Table）是一种数据库对象，数据字典中的表独立于数据库存在，一旦创建，则同样结构的数据库表实体也将在物理数据库中创建。

4．创建数据库表时应注意哪些事项？

■ 有 CURR 类型的字段必须有参照表和参照字段。
■ 建表时主键必须连续排序，中间不能有非主键存在。
■ 插入第一条数据时操作比较特殊

5．列举 ABAP 数据字典中的对象及作用有哪些？

■ Tables。
■ Views。
■ Data elements。
■ Structure。
■ Domains。
■ Search Helps。
■ Local Objects。

6．CHECK TABLE 和 VALUE TABLE 各指的是什么？关系怎么样？

如果一个表字段定义了外部关键字，则该表称为外部关键字表，而对应的参照表则称为 CHECK TABLE。如果在数据字典中为一个表字段指定了约束表，则其输入帮助可以是该表中的关键字段值列表，或者是约束表中对应自动制定的输入帮助。

■ CHECK TABLE 用于字段级别的检查。
■ VALUE TABLE 用于域级别的检查。

7．数据库表维护有哪些内容？

■ 表字段。

数据库表与结构类型、表类型的区别：后二者是单纯的数据类型，前者还包含其他相关设定，如外码、索引等。

■ 技术设定

通过 Technical Settings 按钮维护，主要维护定义表中所存储数据的类型等信息。例如操作数据库表时是否需要进行缓冲（对不需经常更新的数据库表提高后续查询效率）、是否要创

建系统日志等。

■ 索引——仅包括部分字段的数据库表拷贝，通过 Indexes 按钮维护。
  - 主索引：只包含表关键字和指向整个数据条目的指针，系统自动生成。
  - 附属索引：根据其他字段生成索引，按指定方式排序，自行创建，协助主索引提高搜索效率。

5.3 报表相关问题

5.3.1 基本报表

1．报表事件

1）过程化事件有哪些？

■ INITIALIZATION。

■ START-OF-SELECTION。

■ END-OF-SELECTION。

2）选择屏幕事件有哪些？

■ AT SELECTION-SCREEN OUTPUT。

■ AT SELECTION-SCREEN。

■ AT SELECTION-SCREEN ON field。

■ AT SELECTION-SCREEN ON BLOCK block。

■ AT SELECTION-SCREEN ON RADIOBUTTON GROUP group1。

■ AT SELECTION-SCREEN ON seltab。

■ AT SELECTION-SCREEN ON END OF seltab。

■ AT SELECTION-SCREEN ON HELP-REQUEST FOR field。

■ AT SELECTION-SCREEN ON VALUE-REQUEST FOR field。

3）在使用逻辑数据库时，如 HR 报表，涉及的逻辑数据库事件有哪些？

■ GET node。

■ GET node LATE。

4）报表操作事件有哪些？

■ TOP-OF-PAGE。

■ END-OF-PAGE。

■ AT-USER-COMMAND。

5）交互式报表事件有哪些？

■ AT USER-COMMAND。

■ AT LINE-SELECTION。

■ AT PF-FUNCTION KEY（设置功能键）。

6）按顺序举例说明各个 Report 事件及其作用？

■ LOAD-OF-PROGRAM ：程序执行时，自动触发该 EVENT。

■ INITIALIZTION：对程序变量进行初始化。

- ■ AT SELECTION-SCREEN OUTPUT：选择屏幕 PBO 事件，显示选择屏幕之前触发。
- ■ AT SELECTION-SCREEN：选择屏幕 PAI 事件，用户在选择屏幕上执行某些功能。
- ■ START-OF-SELECTION：选择开始事件。
- ■ END-OF-SELECTION：选择结束事件，一般在该事件中输出列表。
- ■ TOP-OF-PAGE：页眉控制。
- ■ END-OF-PAGE：页角控制。
- ■ AT LINE-SELECTION：行选择控制，用户选择某列表行时触发。
- ■ AT USER-COMMAND：用户交互控制，用户选择某个 Gui 功能时触发。

2．报表选择画面

1）报表的选择画面上能否添加自定义工具栏？

能够添加，通过语句 SELECTION-SCREEN FUNCTION KEY n（n 为 1 至 5，最多定义 5 个）。

2）如何对报表的选择画面元素进行分组？

在 PARAMETERS 或 SELECT-OPTIONS 后使用附加语句“MODIF ID + 组名”。

3）如何更改屏幕各元素的状态？举例一些比较常用的屏幕属性。

在报表 AT SELECTION-SCREEN OUTPUT 事件中或 PBO（PROCESS BEFORE OUTPUT）中，利用如下代码：

```
LOOP AT SCREEN.
    ….MODIFY SCREEN.
ENDLOOP.
```

SCREEN-NAME：画面元素的名称。

SCREEN-GROUP1 ～ SCREEN-GROUP4：对画面元素的分组。

SCREEN-INPUT：能否输入。

SCREEN-INVISIBLE：是否可见。

SCREEN-LENGTH：可见长度。

SCREEN-ACTIVE：是否是可用的状态。

4）在什么时候触发 TOP OF PAGE 语句？

- ■ 第一个 WRITE 命令。
- ■ 输出行数超过 LINE-COUNT 属性值。
- ■ NEW-PAGE 命令。

5）Report 的跳转语句是什么？PICK 是什么？怎么用？

该语句为 AT LINE-SELECTION。代码 PICK 为内部触发 AT LINE-SELECTION 事件。在预定义界面中，将“编辑→选择”和〈F2〉分配到 PICK。设置功能键〈F2〉自动激活鼠标双击功能。

6）介绍一下下拉列表常用到的技术点？

下拉列表多用到 HIDE 指令、READ LINE 指令、事件 AT LINE-SELETION、AT USER-COMMAND 及 AT PF<N>。事件 AT LINE-SELECTION 定义用户双击列表某行时触发的事件，AT USER-COMMAND 定义用户按下某按钮，或者选择某一菜单项或输入命令后触

发的事件及 AT PF<N>用户按下功能触发的事件。

7）如果运行报表输出的程序，不想输出到屏幕上，而是直接打印应该怎么做？

在程序中利用代码 NEW-PAGE PRINT ON 将生成的列表直接发送到 SPOOL 系统进行打印，不在屏幕上显示。

5.3.2 ALV报表

1．ALV 报表实现的流程是怎样的？

声明数据对象 → 定义选择画面 → 从数据库取数 → 处理数据 → 调用显示数据。

2．显示 ALV 常用的几个函数是什么？

■ REUSE_ALV_GRID_DISPLAY。

■ REUSE_ALV_LIST_DISPLAY。

■ REUSE_ALV_GRID_DISPLAY_LVC。

3．ALV 显示中的小计如何实现？

■ 使用 SUBTOT：

ALV 可以实现自动小计，首先字段目录参数中添加 DO_SUM = X，然后传入排序表，可以在排序表中指定需要小计的列。

■ 使用 AT 语句：

在 LOOP 循环中使用 AT END OF FIELD 逻辑。

4．函数 ALV 和面向对象 ALV 有什么不同？

函数 ALV 和 面向对象 ALV 都能够实现按钮自定义、数据修改、按钮处理自定义等操作。通常情况下函数 ALV 主要用于报表数据展示及简单交互；面向对象 ALV 主要用于 Dialog 程序开发，可以进行复杂的控制，比如单元格的修改控制（函数只能控制到列修改性）、自定义〈F4〉等。面向对象 ALV 可以根据容器排列很方便的定义布局，一个屏幕可以放多个 ALV，但是函数 ALV 只能一屏显示一个 ALV。

5．ALV 是什么，能简单介绍一下 ALV 的作用吗？

ALV（ABAP LIST VIEW）是 ABAP 的报表显示工具，它把内表中的内容以设定的格式显示出来，附加一些功能，如下载为本地文件等。

5.3.3 SmartForm报表

1．谈谈 SmartForm 中，Template 和 Table 表格的区别？

Template 是静态表格，单元格的输出是一定的。Table 是动态表格，包含在循环内，内表能够按条输出。

2．SmartForm 如何实现公司 LOGO 打印？

■ 用 SE78 导入公司 LOGO，导入时只能导入位图和 GIF 文件。导入位图时，要注意用 256 色，否则会导致图片打印有问题。然后在 SmartForm 中添加图片即可。

3．SmartForm 中如何控制段落、单个字符的输出格式？

用 Styles 可以定义不同的段落、字符格式，在文本输出时可以指定格式。

4．什么是 SmartForm，简述其创建过程？

SmartForm 的基本思想是为了提高处理的效率，将数据筛选、整理过程与数据在 Form 中

的应用处理及格式化过程相分离。

将数据库中的数据通过 SmartForms 格式化并输出的步骤如下：

- 提取应用数据。
- 定义 Form 格式及相关数据接口。
- 将数据传输至 Form 中，并批量输出。

5.4 对话程序

5.4.1 事件

1．Dialog 中的事件有哪些？

- PBO（PROCESS BEFORE OUTPUT）。
- PAI（PROCESS AFTER INPUT）。
- POV（PROCESS ON VALUE-REQUEST）。
- POH（PROCESS ON HELP-REQUEST）。

2．介绍一下 Dynpro 程序结构以及运行机制？

- 逻辑上分为 PBO，PAI 以及 POV，POH
- 建立程序时物理上 TOP，O01，I01，F01.
- 运行机制：TCODE->PBO->PAI……

3．解释一下 PBO，PAI 以及 POV，POH？

- PBO：PROCESS BEFORE OUTPUT（显示前处理过程），在向用户显示屏幕 Gui 之前触发，一般做画面初始化和设定 Status、Title、动态设定画面属性和清空变量。
- PAI：PROCESS AFTER INPUT（输入后处理过程） 在用户进行某些屏幕行为后触发，对应一些用户操作、画面检查。
- POV：PROCESS ON VALUE-REQUEST 对应功能键〈F4〉，用户选中画面上某个字段按〈F4〉键，类似检索帮助。
- POH：PROCESS ON HELP-REQUEST 对应功能键〈F1〉。

5.4.2 控件

1．如何在 Table Control 中实现选中一行或多行的效果？

将内表第一个字段定义为一个长度为 1，类型为 C 的变量，并在 Table Control 属性框中将该字段填写到选择字段属性中。

2．Dialog 开发的几个常用控件是什么？

子屏幕、文本、输入框、输出框、容器、表控制、按钮等。

3．简述表格控制和步循环的区别？

表格控制和步循环是用于屏幕表格显示的对象。表格控制只是增强了的步循环，它可以使用桌面应用程序中表格工具的“Look”和“Feel”来显示数据。表格控制还提供使表格易于查看和使用的专用格式化功能。

步循环的一个特点是它们的表行能够在屏幕上跨越多行。相反，表格控制中的行总是单

行，但可以很长（表格控制的行能够滚动）。

通常表格控制所提供的许多特征由系统的 SAPGUI 前端操纵，因此，不必在 ABAP/4 事务中编制任何特征。

5.4.3 语法

1．Dynpro 中画面调用的几种方式和各自的好处是什么？

- SET SCREEN next_screen_number：为当前屏幕指定后续屏幕，不会自动返回当前屏幕。
- CALL SCREEN screen_number 及 CALL TRANCTION tcode：挂起当前屏幕序列，进入新屏幕（序列）或事务。在新的屏幕序列中，通过语句 LEAVE SCREEN TO SCREEN 0 调用的屏幕就能够返回到挂起的屏幕序列上（返回到原程序）。
- LEAVE SCREEN：如果未完成当前屏幕控制流就要转入后续屏幕，使用此语句。
- LEAVE TO SCREEN screen_number 及 LEAVE TO TRANCTION tcode：未完成当前屏幕就要转入后续屏幕，并且不返回原有程序。

2．举例说明 Dynpro 的动态属性有哪些？

- NAME 屏幕字段名称。
- GROUP1、2、3、4，屏幕字段属于哪个组。
- ACTIVE 1 可用，0 不可用。
- REQUIRED 1 必入力，0 非必入力。
- INPUT1 可入力，0 不可入力。
- OUTPUT1 可显示，0 不可显示。
- INTENSIFIED1 强调显示，0 非强调显示。
- INVISIBLE1 为*，0 为输入即可见。
- LENGTH 字段输出长度。
- DISPLAY_3D 以三维框架显示字段。
- VALUE_HELP 有查询帮助的字段。

3．Dynpro 逻辑块中，Field 的作用是什么？

- 用来传值。
- 进行字段输入检查（防止所有控件都变灰）。

4．Field Module 和 On Value-Request 有什么作用？

在 Dynpro 程序中，控制屏幕控件是否输入修改值等操作，主要用于屏幕数据的检查。

5.5 批处理

5.5.1 原理

1．BDC 录屏的事务代码是什么？

SHDB。

2．简述 BDC 与 BAPI 之间的区别？

都可以用作数据导入，但是 BDC 是完全模拟前台操作，而 BAPI 是 SAP 提供的标准 API，

通过调用 BAPI 在后台生成相应数据，性能高于 BDC 导入方式。

3．BDC 录制的注意事项？

将要填写的字段全部输入，保证其能正确被录制下来，不需要填写的字段不要操作，以免录制后产生冗余字段而产生错误。

4．谈谈 BDC 的运行模式和更新模式？

运行模式包括：A.全屏幕显示；E.只有在发生错误时显示屏幕；N.不进行屏幕的显示，将相关信息记录到信息内表里；P. 不进行屏幕的显示，可进行 DEBUG 测试。

更新模式包括：A.异步更新；S.同步更新；L.本地更新。

5．BDC 执行出错时，为查看设置错误处，显示模式应怎样设置，A、E、N 分别代表什么含义？

可以设置 E.出错时转前台运行，有机会输入修正数据。

如果不想马上转前台用 Insert_Group 插入一个 Session。

A .全画面表示；

E.碰到错误时弹出；

N.不弹出错误画面，后台运行。

6. 什么是 BDC（Batch Data Communication）？简单介绍其流程？BDC 调查有哪些方式？BDC 有几种，分别是什么？

BDC 是 SAP 常用的一种数据传输方法。用于一些数据量大，但是对速度又要求不高的数据传输。

其流程大致分为如下几步：

- 提炼数据。
- 把提炼的数据转换到 BDCDATA 内表中。
- 根据 BDC 类型通过 Call Transaction 形式执行，或通过 Session 形式执行，如果 Session 执行成功，数据将被传送成功。

调查 BDC 的事务代码：SHDB，查看 Session 执行结果的事务代码：SM35。

7．如果想把一个 BDC 的程序定义为零点开始启动应该怎么做？

通过计划后台作业（事务代码 SM36），定义 Job 并设定 Job 起始时间为零点。

5.5.2 语法

1．在 BDC 程序中，BDC 内表所用的构造名及其字段是什么？

BDC 内表所用的构造名为 BDCDATA，其中字段分别是：

- Program　　Program name of Transaction。
- Dynpro　　Screen number of Transaction。
- Dynbegin　　Indicator for new Screen。
- Fnam　　Name of Database Field from Screen。
- Fval　　Value to Submit to Field。

2．BDC Session 如何创建？在程序中如何处理？

BDC Session 创建过程如下：

- CALL FUNCTION‘BDC_OPEN_GROUP’。

- BDC 内表做成。
- CALL FUNCTION‘BDC_INSERT。
- CALL FUNCTION‘BDC_CLOSE_GROUP。

使用如下语句可以立即执行 BATCH INPUT SESSION。

```
SUBMIT RSBDCBTC AND RETURN WITH QUEUE-ID = WK_QID.
```

在 SM35 下查看 BATCH INPUT SESSION 执行结果。

3．BDC 时如何确定表控件里各行的数据？

用括号数字确定 BDC 输入行数。

5.6 增强

1．什么叫增强？

增强就是 ERP 系统中标准程序的出口，在该出口中由用户根据企业实际需求编写客户化逻辑代码。为什么会有出口呢？因为正常的业务系统不能满足实际需求，这时可以在出口中增加一些功能，来达到要求，这也就是为什么它还叫增强。在 SAP 中，出口就是标准程序最后要调用的一个函数或者功能模块，这个函数或功能模块的输入已经由 SAP 定义好。自己写的 SAP 出口程序在系统升级的时候会被保留，而更改标准程序则会在 SAP 系统升级的时候被覆盖。

2．如何查找 SAP 的第一类增强（User Exit）？

通过 SE37，以 EXIT*开头，找到的函数大多是做系统预留的出口函数。前面说过，用户出口是标准程序留给用户的接口，标准程序通常不允许用户任意修改，如果修改需要申请 Access Key。而且修改标准程序可能导致的错误，ERP 公司通常是不负责的。在 SAP 中，自定义的程序通常以字母 Y 或 Z 开头，因此，出口函数中都预包含了一个 Z 开头的程序。

3．如何查找 SAP 的第二类增强（Customer exit）？其包含哪几方面？

客户自定义程序通过 Call Function 来调用 SAP 库中的一些系统函数，而用户增强的概念刚好与之相反，可以使用一个由 SAP 功能模块调用的功能模块出口。

可以在下面这几个层面对模块进行增强：在 ABAP 程序中的 Function Module Exit（E 类）；在 Gui 接口中的 Menu Exit（C 类）；在指定区域插入一个子窗口的 Screen Exit（S 类）；在 ABAP 数据字典中表和结构的增强 Table Enhancement（T 类）；还有在屏幕指定区域添加用户自定义功能的 Field Exit 和 Keyword Exit。

4．什么是 BADI？

即 Business Add-in，是基于 ABAP 对象的一种增强技术。SAP 预定了一些接口，供客户进行实现，完成增强。它是一种新的功能增强概念，使用 ABAP 对象技术。实现 BADI 要用到类、接口及方法等面向对象的概念。

在进行增强时，必须先定义 BADI，为 BADI 创建一个接口。接着创建一个适配器类（Adapter Class）来实现这个接口，然后创建这个适配器类的实例。

5．SAP 的第四类增强（Enhancement Spot）指什么？

可称为隐式增强，在代码中，可插入增强点对代码进行增强。Enhancement Spot 分为

ENHANCEMENT-POINT 与 ENHANCEMENT-SECTION，ENHANCEMENT-POINT 添加的新代码，能够与旧代码一起运行；ENHANCEMENT-SECTION 增加的新代码，将覆盖旧代码。

6．如何建立增强？

- 找到增强。
- 建立增强项目。
- 添加增强的组件添加进增强项目。
- 编辑增强，添加自己的代码。
- 激活增强代码。

7．与增强相关的事务代码有哪些？

- 对于 Customer Exit：SMOD 查看增强组件，CMOD 建立项目后，实现增强。
- 对于 BADI：SE18 查找接口，SE19 对接口进行实现。

8．如何进行数据库表字段的增强？Append 和 Include 的方式有何区别？

Append 结构（Append Structure）是在向表尾添加字段时创建的，自定义 Include（Customizing Include）由 SAP 开发人员指定，以使用户可以创建新字段。

它们之间的区别在于 Append 结构只能由该表使用，其他表不可使用；Include 能够插入任意一个结构或表，其中当插入表时，该表会自动转换为相同字段的结构。

9．实现功能模块出口时 SAP，应使用应用程序中的哪条语句？

某些 SAP 应用程序中存在功能模块出口，它使用户能够向 SAP 程序中添加一些功能。通过搜索 "CALL CUSTOMER" 可以发现是否存在功能模块出口。

10．什么叫 BTE （Business Transaction Event），BTE 的两种类型？

BTE 可以通过函数的模式附加一个组件到 SAP 系统中，从本质上讲，BTE 也是预定义的访问接口。

BTE 的两种类型：

- Publish and Subscribe Interface：通过事件驱动的模式从 SAP 提供的参数触发一个或多个事件（函数），但是这些事件不会返回信息给 SAP，这种模式不影响 SAP 的正常业务流程。
- Process Interface：属于 SAP 控制业务流程的一部分，这个接口可能对应多个事件，但是每次运行的时候只能根据需要选择运行其中的一个事件，SAP 供给参数接口，并且返回参数给 SAP，这种模式影响了 SAP 的正常业务流程。

5.7 接口

1．SAP 有哪些外部接口及其内容？

SAP 接口模式主要有 RFC、BAPI、IDOC。

- RFC：RFC 是 SAP 系统和其他（SAP 或者非 SAP）系统间的一个重要而常用的双向接口技术，也被视为 SAP 与外部通信的基本协议。简单地说，RFC 过程就是系统调用当前系统外的程序模块，从而实现某个功能，而且调用系统和被调用系统中至少有一个必须是 SAP ABAP 系统。这种远程功能调用也可在同一系统内部进行（如本地 SAP 系统内的远程调用），但通常情况下，调用程序和被调用程序处于不同系统中。

- BAPI：BAPI 实质上是一种特殊的、支持远程调用的业务对象方法，并通过远程支持的功能模块实现。BAPI 支持同步与异步的数据通信过程，典型的 BAPI 功能包括业务对象的创建、属性的读取与更新操作。
- IDOC：IDOC 是 SAP 系统之间或 SAP 系统与外部系统之间电子数据交换的标准数据格式。通过 IDOC 接口可以实现 SAP 系统之间以及 SAP 系统与其他系统之间的数据交换。使用 IDOC 的应用必须能够写入或读取 IDOC 格式的数据。

2．简述 ALE/EDI 接口类型概念及应用？

ALE：用于同一个企业中不同 SAP 系统之间的数据交换，通过 IDOC 格式的数据创建分布式的系统。

EDI（Electronic Data Exchange，电子数据交换）：用于实现不同企业间的电子数据交换，是各应用系统间的结构化电子数据交换标准。EDI 中的 IDOC 符合 EDI 标准，SAP 格式的数据通过 EDI 子系统转换为 EDI 标准，然后与业务伙伴进行数据交换。

ALE/EDI 通过业务连接器（Business Connector），可以将业务文档以 IDOC 的形式发送至 Internet；通过 IDOC 也可连接其他业务应用系统，如 PC 应用、外部工作流工具等。

5.7.1 RFC接口

1．什么是 RFC，有哪些通信模式？

RFC 是 Remote Function Call 的简称，是 SAP 系统和其他系统之间重要而常用的双向接口技术，它包含同步 RFC、异步 RFC、事务性 RFC、队列 RFC 和并行 RFC。

2．RFC 中涉及的常用事务代码有哪些？

- SM51：查看当前 SAP 系统中活动的应用服务器，包括 Server_Names、Host name 和 Type（服务类型）。
- SM49：查看、设定并执行外部命令。
- SM59：配置 RFC 连接，其中相同类型的 RFC 目标被组合至同一个目录。这些远程目标的定义将保存在表 RFCDES 中。

3．根据调用的不同，RFC 接口提供了什么样的服务？

- ABAP 程序的调用接口：任何 ABAP 程序都可以使用 CALL FUNCTION...DESTINATION 预计调用远程函数。DESTINATION 参数告诉 SAP 系统被调用的功能将在调用者之外的系统上运行，通过 RFC 接口与远程系统进行通信。如果远程函数来自 SAP 系统，则必须是时间的函数模块，并在功能库中注册为“支持远程调用 REMOTE-FUNCTION-ENABLE”，通信规则都通过 ABAP 的 RFC 接口实现。
- 非 SAP ABAP 程序的调用接口：在 RFC 的实现过程中，如果调用或被调用一方是非 ABAP 程序，则非 ABAP 程序必须以特定的规格进行编程，确保它可作为 RFC 通信的一方。SAP 系统为外部程序提供 RFC 支持接口（RFC-supported interface）和 GUI 支持接口（GUI-supported interface）。这样，在非 SAP 系统中，外部程序就可以调用并执行 SAP 远程函数模块。

4．RFC 接口的具体功能包括哪些？

- 登录并退出远程系统，并执行权限检查。
- 调用并管理远程系统会话所需的通信例程。

■ 将参数转换成远程系统所需的格式。
■ 处理通信过程中的错误。

5．在通过 CALL FUNCTION 语句进行远程功能调用的基本模式有哪些？

■ CALL FUNCTION - DESTINATION：以同步 RFC 方式实现的远程函数调用。如果 DESTINATION 后无其他附加项，则形成同步 RFC 调用，调用程序等待远程调用的结果以继续执行。
■ CALL FUNCTION - STARTING NEW TASK：以异步 RFC 方式实现的远程函数调用。通过 STARTING NEW TASK 附加项形成异步 RFC 调用，调用程序不等待返回结果继续执行，结果将在回调子程序（CallBack SubRoutine）中接收。
■ CALL FUNCTION - IN BACKGROUND TASK：以事务性 RFC 方式实现的远程函数调用。通过 IN BACKGROUND TASK 形成事务性 RFC 调用，远程功能暂不开始执行，等待 COMMIT WORK 语句出现时，一次性执行一个或多个远程功能。

6．怎么创建一个支持远程调用的 RFC？

在 SAP 系统中，RFC 的创建方式与普通函数模块类似，只是编写函数模块时需要在 Attribute 选项卡中将 Processing Type 选项设为 Remote-Enable Module，在传入、传出参数那里需要选择参数为 Passing Value。

7．怎么调用一个 SAP 标准 RFC？

在 SAP 中，功能模块的调用通过 CALL FUNCTION 语句实现。远程功能作为普通函数模块调用的扩展，可通过在 CALL FUNCTION 语句中添加 DESTINATION 子句完成，其语法格式与普通调用完全相同。

8．怎样建立 RFC 程序？RFC 程序传递的参数是传递值还是引用？如何建立函数组？

在 SE37 中建立函数模块的时候，Attributes 选项卡中选中 Remote Function。RFC 程序传递的参数都是值传递（选中 Pass Value 选项）。建立函数组可进入 SE80，选中屏幕左边的 Repository Browser 选项，在第一个输入框中选中函数组，在第二个输入框中输入需建立的函数组名，再按回车键就可以了。或者从 SE37 进入，在菜单上选中 go to→function group →create group。

9．怎么维护 DESTINATION（远程目标）？

SM59 中包含链接和登录远程系统所需的全部参数信息。还可以在远程调用时直接指定当前系统的应用服务器作为 RFC 目标。通过 SM51 查看当前 SAP 系统中活动的应用服务器。

5.7.2　BAPI接口

1．什么是 BAPI？你使用过哪些 BAPI 实现什么功能？

BAPI 业务应用程序接口（Business Application Programming Interface，简称 BAPI）是面向对象程序设计方法中的一组程序接口。它允许程序员通过 SAP 将第三方软件整合成 R/3 专有产品。为了完成一些特殊的商业任务，如上传交易数据等，R/3 系统中内置了 BAPI。

2．什么是业务对象类型？它包含哪些组件？

业务对象类型是业务对象的定义和描述，面向对象构架的实现基础，也就是 SAP 系统中的类，它封装了业务功能和数据。它包含接口、关键字段、属性、方法和事件 5 个业务组件。

3．如何创建一个 BAPI？

- 定义 BAPI Structure （Structure 不能在 BAPI 中重复使用，因为一旦 BAPI 被释放，其 Structure 将被冻结）。
- 创建 FUNCTION MODULE（每个 BAPI 必须有自己的 Function Group，Function Group 属性必须为 RFC）。
- 创建 Business Object（SW01）。
- 使用 BAPI WIZARD 创建 API Method（这样 BAPI 可以被外部程序调用）。
- 释放 BAPI Function Module，释放 Business Object Type，释放 BAPI 作为 BOR 的一种 Method。

4．编写 BAPI 的注意事项有哪些？

BAPI 不能包括 Call Transaction 或 Submit Report 那些会跳转到别的程序的语句；BAPI 的结构中不能使用 Append 或 Include，并且每个 BAPI 的结构都必须新建；BAPI 中不能使用 Dialog，如需进行文件选择或用户交互的也不能在 RFC 中出现。

5．谈谈与 BAPI 相关的事务代码。

- SWO1 业务对象创建器。
- SWO2 业务对象浏览器。
- SWO3 业务对象仓库浏览器。
- BAPI 对象浏览器。

6．RFC 和 BAPI 的区别？

- BAPI 和 RFC 不是同一个层次上概念。BAPI 是 SAP 提供的基于业务对象的函数，关键是它们处理的对象是 R/3 的业务相关 Business Object，比如单据类销售订单、组织、公司等，它们是一系列实体。RFC 则是一种系统间通讯的方式（Remote Function Call），一个 BAPI 函数往往是一个 RFC 函数。
- BAPI 是 SAP 里一个很好的思想，把业务对象都对象化了。

5.7.3 其他

1．本地文件上传为 PDF、Excel 文件使用的函数是什么？

- KCD_EXCEL_OLE_TO_INT_CONVERT - 上传 EXCEL 文件。
- CONVERT_OTFSPOOLJOB_2_PDF - Converts a OTF spool to PDF。
- CONVERT_ABAPSPOOLJOB_2_PDF -Converts ABAP spool output to PDF。

2．FTP 文件的上传、下载操作是什么？

FTP 程序大致流程为：

- FTP_CONNECT – 打开并登录 FTP 服务器的连接。
- FTP_COMMAND – 在 FTP 服务器上执行一个命令。
- FTP_DISCONNECT –关闭指向 FTP 服务器的连接。

FTP_COMMAND 中输入标准的 FTP 命令。如更改 FTP 目录 "cd dir"、更改本地目录 "lcd localdir"、上传文件"put file"。

3．本地文件上传为 CVS 文件的函数？

- TEXT_CONVERT_CSV_TO_SAP：导入 CSV 数据到 SAP 系统。
- GUI_UPLOAD：文件上传。

4．SAP 文件操作都有哪几种？分别有什么操作？版本有什么限制？

SAP 文件操作的有以下两种文件：

- 顺次文件（服务器文件）。
- 本地文件（客户端文件）。

顺次文件的操作包括：打开、读取、写入、关闭、删除。

本地文件的操作：上传、下载。根据版本具体函数如下：

- 4．7 版本以后：
  - GUI_DOWNLOAD
  - GUI_UPLOAD
- 4．6 版本以前，性能不好：
  - WS_DOWNLOAD
  - WS_UPLOAD
  - DOWNLOAD
  - UPLOAD

5．在文件处理时，文件中出现乱码，应该从哪方面考虑？

Open dataset 时指定字符编码。

6．简单描述 Web Service 的创建过程？

- 发布大体步骤：SE37 向导发布 Web Service, SOAMANAGER 配置端口。
- 调用大体步骤：SE80 创建代理类，LOCONFIG 创建逻辑端口，SE38 程序中调用，创建代理类对象，调用对象的方法。

7．现阶段接口开发使用的技术？

企业服务总线（ESB：Enterprise Service Bus）+WebService。

8．为一个远程调用函数（Remote-Enabled Module）创建 WEB SERVICE 都有什么相关的事务代码？

创建 WEB SERVICE：SE37/SE80。

9．在企业信息系统结构内企业服务总线是如何使用的？

企业服务总线来管理 ERP 和其他接口交互的信息，可以查看监听管理双发系统的交互信息。

5.8 数据库知识

5.8.1 基础知识

1．ABAP 数据字典有哪些对象或元素？

Data element、Domain、structure、lock object、views。

2．据库提交确认和数据库回滚取消语句？

- COMMIT WORK。
- ROLLBACK WORK。

3．什么是 LUW？

LUW 称为逻辑工作单位，同一个逻辑工作单位共享一段内存，将操作放置在同一个 LUW 中可以控制事务的同时提交和回滚。

4．简述 Modify、Insert、Update 对数据库表做操作时的影响。

Modify 操作数据库时，可以使用 From 内表或者工作区来进行多条或单条的更新，要求内表或工作区跟数据库表的结构一致。当数据库表中存在重复记录时，执行更新操作，更新的值为内表或工作区的值；当数据库表中不存在记录时，执行插入操作，插入的值为内表或工作区的值。

Insert 操作数据库时，可以用 From 内表或者工作区进行多条或单条的插入，要求内表或工作区跟数据库表的结构一致。当数据库中不存在重复记录时，执行插入操作，插入的值为内表或工作区的值；如果已经存在重复记录，会出现更新异常。

Update 操作数据库时，可以直接 set 来进行单值更新，可以用 From Table 和工作区进行批量更新，要求内表和工作区跟数据库表的结构一致。当存在记录时执行更新操作；当不存在记录时，没有数据会被更新，同时也不会产生异常，sy-subrc = 0。

5．描述域、数据元素、表字段之间的关系。

域为数据字典中最小的单元，数据元素是基于域进行定义的，表字段则是通过数据元素进行定义的。

6．数据字典有几种缓冲方式，适用范围是什么？

Full Buffer（全缓冲），一个表的数据要么全在内存中要么全不在。当表记录很少，访问非常频繁且很少进行修改时使用。

Generic Area Buffer（常规区域缓冲），语言代码相关的表是最常使用的情况，如文本描述表，缓存满足在 Generic Keys 中定义的字段（个数介于 1 个和表主键数减 1 之间）的值的记录。

Single-Record Buffer（单记录缓冲），数据库表记录比较多，并且经常都是读取一条记录，如 select single。

5.8.2 ABAP和数据库

1. ABAP 数据表的主索引是什么？索引的好处与坏处有哪些？建索引的注意事项有哪些？

数据表的主键即是表的主索引；好的索引能加快数据读取的速度但会增加更新数据库表的时间；建立次级索引时应尽量选取那些查询条件经常使用到的字段。

2．ABAP 透明表有哪几种数据类（Data Class）？对数据的存储有什么影响？

数据类主要分为以下三类：

- 主数据类：经常读取，很少修改的数据。
- 业务数据类：经常更新的数据。
- 组织数据类：系统初始化时存在的数据，也很少修改。

数据类决定了数据实际存放的物理区域。

3．SAP 中有几种表，他们的区别是什么？

Transparent table（透明表），Pooled table（共享表）和 Cluster table（簇表）。透明表是和 DB 层的物理表（Physical Table）对应的，后两者则是不对应到 DB 表的。 比如 TBSL 就是

一个共享表，在 DB 层找不到此表。然后多个簇表或共享表组成一个表簇（table cluster）或表池（Table Pool），表簇和表池是 DB 层的一个物理表。

4．什么是簇表（cluster table）？举出知道的簇表。

簇表是逻辑上有关联的几个表，在定义的时候分配给一个表簇，如 Bseg。

5．找数据库表有哪些常用的方法？

■ 单击画面上需要查找的字段，单击〈F1〉键，在弹出画面中的技术信息中查找。

■ 通过 ST05 进行数据库操作的跟踪，对于在前台界面进行数据的新增或更新，在数据库中都会有所体现。

■ 通过事务 SE80→Repository Information System→ABAP 字典→数据库表格中，对某个字段进行查询。

6．如何提高程序效率？

■ 使用 OPEN SQL 时注意以下原则：

● 尽可能减少满足条件的数据条目数量。

● 减少数据的传输量，以减少网络流量。

● 减少访问的数据库表量。

● 减少查询难度，可以通过整理选择标准来实现。

● 减少数据库负载。

■ 不要在 LOOP 中访问数据库，尽量将数据预先提取到内表中，然后再通过语句 READ TABLE WITH KEY ... BINARY SEARCH 将内表进行数据整合。

■ SELECT 语句尽量提取需要的字段，避免使用 SELECT *，对于不需要的字段避免抽取。

SELECT 语句 WHERE 条件，应该先将主键相关条件放在前面，然后按照比较符 = < > <> LIKE IN 的顺序排列 WHERE 条件。

■ 读取内表使用二分查找方式 BINARY SEARCH，读取 TABLE 之前使用 SORT TABLE BY 对内表进行排序。

■ 尽量使用 Select（max，min，sum，avg）、select single、for all enteris、append 和 collect 语句。

■ 避免使用 select distinct，在抽取数据到内表前先 sort，后用 DELETE ADJACENT DUPLICATES 语句来消除重复行，再 delete。

■ 不要使用 SELECT...ENDSELECT 语句。

■ 用 SORT 代替 SELECT...ORDER BY。

■ 尽量多用 WHERE 语句进行条件抽取。

7．表内联（外联）限制条件有哪些？

LEFT OUTER JOIN 限制条件：

■ 在 ON 附加项中，只能使用"="操作符，且必须至少有一个条件是对主选择表和结合表中的字段进行比较。

■ 不能在 WHERE 子句中对结合数据表中的字段进行条件限定，该语句中的 OUTER 关键字可以省略。

INNER JOIN 限制条件：

■ 不同逻辑条件之间只能通过 AND 连接。

■ 每一个条件中必须包含一个主选择表中的字段。

LEFT OUTER JOIN 与 INNER JOIN 的主要区别是 LEFT OUTER JOIN 中对于主选择表的数据，即使在结合数据表中结合条件字段值不存在，也将该数据行选出，结合表中不存在的字段保持空白。

8. UPDATE、DELETE 数据库时的注意事项？

■ DELETE 语句用以删除数据表里记录。

注意：删除记录并不能释放 ORACLE 里被占用的数据块表空间，它只把那些被删除的数据块标成 unused。 如果确实要删除一个大表里的全部记录，可以用 TRUNCATE 命令，它可以释放占用的数据块表空间：TRUNCATE TABLE 表名，且此操作不可逆。

- DELETE FROM dbtab WHERE <cond>只要删除一行，SY-SUBRC 返回 0。
- DELETE dbtab FROM wa 删除成功，SY-SUBRC 返回 0。
- DELETE dbtab [CLIENT SPECIFIED] FROM TABLE itab 如果有一行不能删除，系统继续处理下一行，成功删除内表中所有行时，SY-SUBRC 返回 0。如果内表为空，会删除所有数据，且 SY-SUBRC 和 SY-DBCNT 都返回 0。

■ UPDATE 语句用以修改数据表里记录。

注意：以上 SQL 语句对表都加上了行级锁，确认完成后，必须加上事物处理结束的命令 COMMIT 才能正式生效，否则改变不一定写入数据库里。如果想撤回这些操作，可以用命令 ROLLBACK 复原。

- UPDATE dbtab SET f1 = g1 ... fn = gn WHERE <fix_key>只能更新非关键字，只要更新一行，SY-SUBRC 返回 0。
- UPDATE dbtab FROM wa 根据工作区中的关键字更新对应的条目。
- UPDATE dbtab FROM TABLE itab 根据内表关键字批量更新数据。

9. 在程序中使用逻辑数据库的两种方法是什么？

在程序中使用逻辑数据库有两种方法，通常是通过 GET 事件或者功能模块进行调用，如 NODES node。

通常情况下报表程序中输出的列表数据来自数据库，在 ABAP 程序中，可使用 Open SQL 或者 Native SQL 读取这些数据库表。

逻辑数据库则提供另一种选择数据的方式，可以代替 SQL 语句在数据库中提取数据，供程序操作。因而，可以认为逻辑数据库是一种报表设计程序中的辅助工具，而不能将其误解为一种可进行数据存储的数据库。SAP 中提供多种类型的逻辑数据库，开发人员也可自行定义数据源。它的维护工具的事物代码是 SE36 或 SLDB。

与普通 SQL 语句读取数据相比较，逻辑数据库有以下优点：

■ 预设的选择屏幕。

■ 逻辑数据库中可能包含多个数据表作为节点，在使用逻辑数据库的程序中，程序开发人员可以忽略具体的数据表之间通过外部关键字设定的关系，而是使用 GET 事件按照节点层次逐级获得数据。

- ■ 逻辑数据库可以提高程序数据的筛选效率，并在数据筛选的同时进行用户数据库访问的权限检查。
- ■ 逻辑数据的维护和设定可以独立于程序进行，如果数据源发生变化，也可以直接修改数据库，无需对所有程序进行修改。

一般只有在 HR 模块编程时才需要使用到逻辑数据库。其他模块基本用不到，但 HR 必须用。

5.9 业务知识

1．SAP R/3 包含哪些模块？

- ■ Financial Management （FI，财务会计）。
- ■ Cost Management and Controlling（CO，管理会计）。
- ■ Material Management（MM，物料管理）。
- ■ Production Planning（PP，生产计划和控制）。
- ■ Sales and Distribution （SD，销售和分销）。
- ■ Production Planning（PP，工程/项目管理）。
- ■ Quality Management（QM，质量管理）。
- ■ Project Management（PM，工厂维护与管理）。
- ■ Human Resource（HR，人力资源）。
- ■ Enterprise Control（EC，企业控制）。
- ■ Treasury（TR，财务管理）。
- ■ Investment Management（IM，投资管理）。

2．简述 MM、SD、PP 模块标准流程及主要事务码、数据库表？

- ■ MM 模块的标准流程：
  - ● 当库存不足时，由后勤等相关部门提出采购申请。
  - ● 采购部门接到采购申请，经审查合格后向供应商（1 个或多个）询问价格，在对供应商的报价研究后，确定供应商。
  - ● 采购部门向供应商下达采购订单。
  - ● 入库等相关管理人员对供应商送达的商品进行入库确认。
  - ● 采购部门对供应商提出支付请求进行确认，经确认无误后，向财务部门提出支付申请，财务部门对支付申请进行审核合格后，进行支付处理。
- ■ SD 模块的标准流程：
  - ● 销售部门接到客户寻价后，在系统中登录客户的寻价信息。
  - ● 销售部门对登录的寻价信息进行价格确认，在系统中登录报价信息并告知客户。
  - ● 接到客户的采购订单后，销售部门在系统中登录采购订单。
  - ● 仓库管理部门对客户所订购的商品进行筹措完毕后，进行出库处理，并在系统中登录出库单（库存不足，由生产计划部门制定计划，生产部门组织生产）。
  - ● 销售部门向客户提出支付请求，客户支付后，给客户出具发票，在系统中登录开票信息。

■ PP 模块的主要流程：

首先，生产计划部门需要根据客户订单以及在预测等基础上制定工厂的主生产计划（有了需求，才能组织生产）。

- 生产计划部门根据主生产计划（MPS），运行物料需求计划（MRP）（前台或后台执行）。运行 MRP 会产生 2 个结果，对库存不足的原材料生成采购申请，对产成品生成计划订单（对原材料的采购参见 MM 流程）。
- 生产管理部门将计划订单转化为生产订单。
- 生产管理部门对生产部门下达生产订单。
- 仓库管理部门对生产订单发货。
- 工单确认，生产部门对生产完成的各道工序逐个进行确认和报工。
- 仓库管理部门对生产订单（按订单生产的产品）收货。

3．简述 MM、SD、PP、FI 模块数据库表。

■ MM 模块主要数据库表：

- T001（公司代码）。
- T001W（工厂/分支机构）。
- T001L（仓储地点）。
- MARA（物料主数据）。
- LFA1（供应商主数据　（一般地区））。
- MARC（物料的工厂数据）。
- MARD（物料的仓储位置数据）。
- EKKO（采购凭证抬头数据）。
- EKPO（采购凭证项目）。
- EKET（计划协议计划行）。
- MSEG（凭证段：物料）。
- MKPF（抬头：物料凭证）。

■ SD 模块主要数据库表：

- KNA1（客户主数据）。
- VBAK（销售凭证：抬头数据）。
- VBAP（销售凭证：项目数据）。
- VBEP（销售凭证：计划行数据）。
- VBRK（出具发票：抬头数据）。
- VBRP（出具发票：项数据）。

■ PP 模块主要数据库表：

- STKO（BOM 表头）。
- STOP（BOM 项目）。
- CRTS（工作中心能力分配）。
- CRHD（工作中心表头）。

■ FI 模块主要数据库表

- BKPF（会计核算凭证标题）。

- BSEG（会计核算凭证段）。
- BSID（会计核算：客户的第二次索引）。
- BSAD（会计核算：客户的第二个索引（已结算项目））。
- BSIK（会计核算：供应商的第二次索引）。
- BSAK（会计核算：供应商的第二个索引（已结算项目））。
- BSIS（会计核算：用于总账科目的第二个索引）。
- BSAS（会计核算：总账科目的第二个索引（已结算项目））。

4．简述 MM、SD、PP、FI 模块主要事务码。

■ MM 模块主要事务码：
- MMBE　库存总览。
- ME51N 采购申请。
- ME21N 采购订单。
- MIGO　收货。
- MIRO　后勤发票校验。

■ SD 模块主要事务码：
- VA11　 询价。
- VA21　 报价。
- VA01　 销售订单。
- VL01N 出库。
- VF01　开具发票。

■ PP 模块主要事务码：
- CS01 创建 BOM。
- CS07 将 BOM 分配给工厂。
- CR01 创建工作中心。
- CA01 创建工艺路线。
- MD61 创建独立需求计划。
- MD05 显示 MRP 清单。
- MD04 库存需求订单。
- MD02 运行物料需求计划。
- COOIS 显示生产订单清单。
- CO02 更改生产订单（下达生产订单）。
- MD42 对成品的生产计划。

■ FI 模块主要事务码：
- FB50 输入总账科目凭证。
- FB60 供应商凭证。
- FB70 客户凭证。
- FB02 总账科目过账。
- F-04 过账并清账。
- F-28 收款。

● F-53 付款。

5．简述 HR 模块各功能模块、主要事务代码及表。

HR 模块主要包括组织管理、人事管理、时间管理、薪酬管理，覆盖了 HR 活动中组织、招聘、发工资、任职、考勤、培训、提升、降级、辞退等各个事项。

■ 主要事务代码如下：
  ● PA10 人事档案。
  ● PA20 显示人力资源主数据。
  ● PA30 维护人力资源主数据。
  ● PA40 人事活动维护事务。
  ● PA41 变更输入/离开日期。
■ 主要表如下：
  ● T001P 人事领域/sub 领域。
  ● T500L HR 国家类别。
  ● T500P 人事领域。
  ● T510A HR 工资表类型。

5.10 其他

1．简述 SAP 系统中程序事件和系统事件。

■ 程序事件需要调用 Dispatch，由系统工程师决定这个事件在什么时间响应，而系统事件不需要调用 Dispatch，什么时候响应系统事件由系统决定（自动调用 Dispatch）。
■ 程序事件在 PAI 时，可以访问屏幕上面的值，而系统事件不能。系统会重写屏幕上的值。
■ 程序事件在用户误操作的情况下可能丢失。

2．如何建立一个外部数据库的连接？

事务代码 DBCO 或者维护表 DBCON。

3．SAP 包括哪些传输技术？

■ SAP 标准数据传输程序，包括批输入、调用事物、直接输入等数据传输技术。
■ 通过 BAPI 和 IDoc 传输数据。
■ 自定义数据传输程序。

4．如何将内表数据转换成一个 XML 文件？

可以使用普通的方法（SAP 提供的一些生成 XML 类 IF_IXML，IF_IXML_ENCODING，IF_IXML_COMMENT，IF_IXML_ELEMENT，IF_IXML_ATTRIBUTE，IF_IXML_OSTREAM，等）

通过事务代码 STRANS 来创建一个 Transformation，在这里定义好 XML 的格式、内容等，然后通过 CALL TRANSFORMATION xxx OPTIONS xml_header = 'no'SOURCE root = itab header = header RESULT XML xml_string 来将内表数据转换为 XML 文件。注意这个内表不能使用带表头的内表。

5．怎么导出一个 WSDL 文件？

根据 SAP 版本不同，分别使用 SOAMANAGER（基于 NETWEAVER）/WSADMIN 和 WSCONFIG（Release Web Services （基于 R3 的）for SOAP Runtime）导出 WSDL 文件。

6．怎么导入一个 WSDL 文件？

在 SE80 里面导入该 WSDL 文件，导入后，会生成相应的 Proxy Name（Class Interface），在这个类（Class）里面就会包含该 WSDL 文件提供的方法。然后根据需要，在 SM59 创建一个远程连接，在 LPCONFIG 里面创建逻辑端口（指定端口的目的地）。最后在接口程序里面使用相应的类。

7．创建后台程序的 TCODE 是什么，如何在程序里面调用后台程序？

- TCODE：SM36。
- 调用程序的函数：JOB_OPEN 和 JOB_CLOSE。

8．什么是授权对象？在 ABAP 程序中使用哪条语句进行授权检查？

授权对象由一组字段组成，这些字段中的值将被用于进行授权检查。ABAP 程序中使用 AUTHORITY-CHECK 语句根据授权对象进行授权检查。在 AUTHORITY-CHECK 语句中，必须指明授权对象的所有字段，但有一个例外，可以用 DUMMY 关键字来绕过某个字段的检查。

一个授权对象中最多可以定义 10 个字段。

9．在 SAP 系统中是怎样定义“修改”的？它们对更新（Upgrade）有怎样的影响？

修改是指用户对 SAP 发布的库对象（Repository Object）进行的更改。必须在更新期间对修改进行评审（Review），来决定是否应该使用新的 SAP 对象，以及将来使用时是否需要进一步修改该对象。

10．什么是事务变式？为何要使用它？

事务变式是一组屏幕变式，用于预定义屏幕行为和默认值。通过使用变式功能，可以将用户不需要的字段、子屏幕及全屏幕从用户视图中取消。可以给任何输入字段设置默认值，字段也可以不用带 Ready for Input 状态。

只能为对话和报表事务创建事务变式，变式中只能包含普通屏幕、子屏幕及对话屏幕。

开发人员可以使用 GuiXT 脚本语言通过事务变式维护对屏幕进行修改。修改屏幕布局的方式有：插入按钮、值帮助（Value Helps）、移动对象、插入屏幕等。

11．请列出用户修改 SAP 标准功能的不同方式。

SAP Standard 可以通过 Personalization、Customizing、Modifications、Enhancements 及自定义 ABAP 程序进行更改。这些方式的示例如下：

- Personalization：Personalization 技术包含创建变式、设置/获取参数及活动组（Activity Groups）。
- Customizing：是最常用的使用 SAP 工具（如 R/3 Reference Model and Implementation Guide）更改 SAP Standard 的方式。可以认为 Customizing 是实施 R/3 所必需的，通常由 Functional Team 来执行。
- Enhancements：常由开发 Team 执行，包含的活动有：字典增强、Funciton Module Exits、菜单和屏幕出口及 Business Add-ins （BADI）。

■ 自定义 ABAP 程序：可以与 SAP 对象或自定义开发的对象一起工作。
■ Modifications：不建议对 SAP 对象使用 Modifications。使用 SSCR （SAP Software Change Registration）注册所有对 SAP 对象的手动修改。

12．是否使用过 QUERY，说明过程。用到哪些 TCODE？

■ 建 SAP QUERY 的过程主要有三步：
  ● 创建用户组。
  ● 创建信息集。
  ● 创建 SAP QUERY。
■ 与这三步对应的事务代码分别是：
  ● SQ01：SAP QUERY。
  ● SQ02：InfoSet。
  ● SQ03：UserGroup。

13．简述在 SAP 中如何进行 Job 管理？

（1）后台处理中，R/3 可以自动运行一个类型的报表程序（自定义屏幕除外）或外部程序。

（2）后台处理步骤：

1）用户通过规划后台处理系统中的作业，通知 R/3 系统所需要进行的工作。设定事件、程序、变式、打印等。

2）到达指定时间后，后台处理系统将启动作业并运行指定的程序。

3）稍后，用户可以检查工作是否完成，显示跟踪该作业执行的日志，查看后台作业的状态。

（3）优点：

1）实现并行，不影响用户操作。

2）可定期自动重复作业，节省人力。

3）晚上执行可降低系统负载。

（4）在 ABAP 中提交作业，将列表发送至 Spool（假脱机）系统。

1）通过功能模块 JOB_SUBMIT 将程序直接发送至后台。

2）使用 SUBMIT 语句中的 VIA JOB 附加项，WITH 附加项则可以提交选择屏幕参数或 USING SELECTION-SET 可指定变式，同时，必须指定 AND RETURN 选项。

```
SUBMIT rep ... USER user VIA JOB job NUMBER n [... WITH p1 = f1... | USING SELECTION-SET vari] ... AND RETURN.
```

（5）已经规划的作业在释放之前不能运行。

（6）显示作业状态：SM37。

14．如果发现报表性能问题，应如何去处理？

首先要检查代码，检查代码是不是有误，看看能不能进行常做的代码优化，查找耗费资源的原因。

如果是数据太多，可以增加筛选数据的条件，如在画面上增加查询条件。利用事务 SE30 定位错误，有一个运行分析图示，找到瓶颈，确认是数据库操作还是代码运行的问题。如果

是代码运行的时间较长，把标准内表类型改成 Sorted 或者 Hash 的内表。

15．定义函数所涉及的属性、参数及其作用。

事务 SE37 定义功能组添加函数时涉及参数如下：

- Attribute：设定功能模块类型、名称等通用属性。
- Import：定义输入参数，从程序传递到功能模块。
- Export：定义输出参数，从功能模块传递到程序中的参数。
- Tables：定义内表参数，将内表整体输入、输出。
- Exceptions：定义可能出现的异常。
- Changing：定义输入、输出参数。

第 6 章　常 用 工 具

6.1　SAP 简单报表生成工具

SAP 系统提供了两个列表生成工具：Quick Viewer 和 Quick Query。见到这两个工具，关键用户如获至宝，例如：关键用户可以创建“固定资产列表”“物料列表”“客户列表”等。

Quick Viewer 与 Quick Query 的区别在于 Quick Viewer 不能创建包含统计、排序等功能的列表，只能创建基本列表。使用这两个工具只需要少量的数据库知识。首先，顾问需要明白信息系统的数据保存在数据库表中。其次，顾问需要知道关系数据库的根本就是表和表之间的关系、约束，这样才能组织数据源（表的外联和内联）。

6.1.1　Quick View

Quick View 访问路径：

菜单路径：工具→ABAP 工作台→实用程序→快速浏览器。

事务代码：SQVI。

1．Quick View的作成

步骤一：Quick View（第一画面）。

在快速浏览的栏位填入 Quick View 的 ID，单击 创建 按钮，如图 6-1 所示。

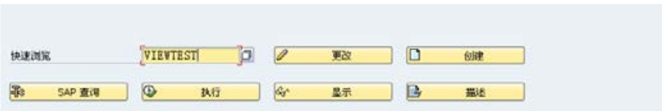


图 6-1

步骤二：Quick View（编辑画面）。

填写 QuickView 所在的标题和注释，指定数据源及显示模式，如图 6-2 所示。



图 6-2

注：数据源可为一个表、一个逻辑数据库、一个连接或来自 SAP 查询的一个信息集，在这里选择表连接（即：从多个关联表中取数）。

单击图 6–2 中的按钮。

步骤三：Quick View（编辑画面）。

定义列表的字段选择和排序，以及列表运行前的选择屏幕等信息。

“可用域”包括了上一步选择的数据库表中的所有字段。“字段中清单”包括在列表中将会显示的字段。根据需要将字段从“可用域”移至“字段中清单”。通过按钮和按钮，调整选择字段的输出顺序，如图 6–3 所示。

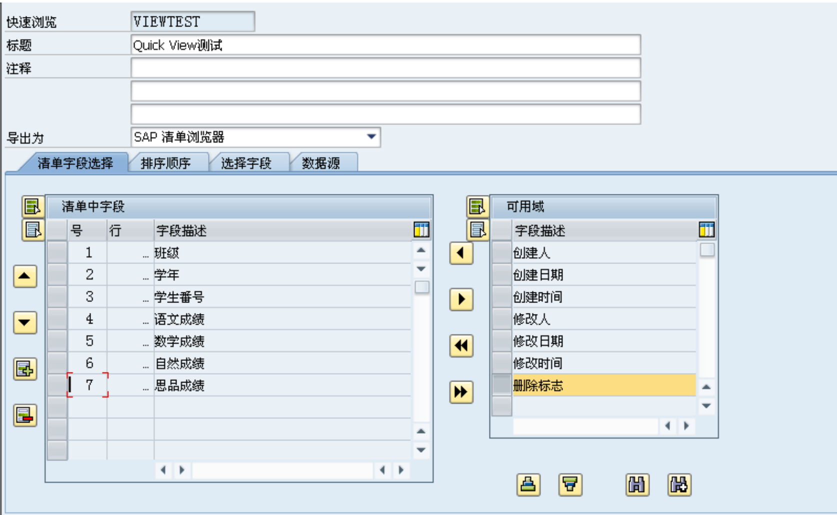


图 6–3

下面是“Sort Sequence”视图，在这里指定列表的排序规则，如图 6–4 所示。

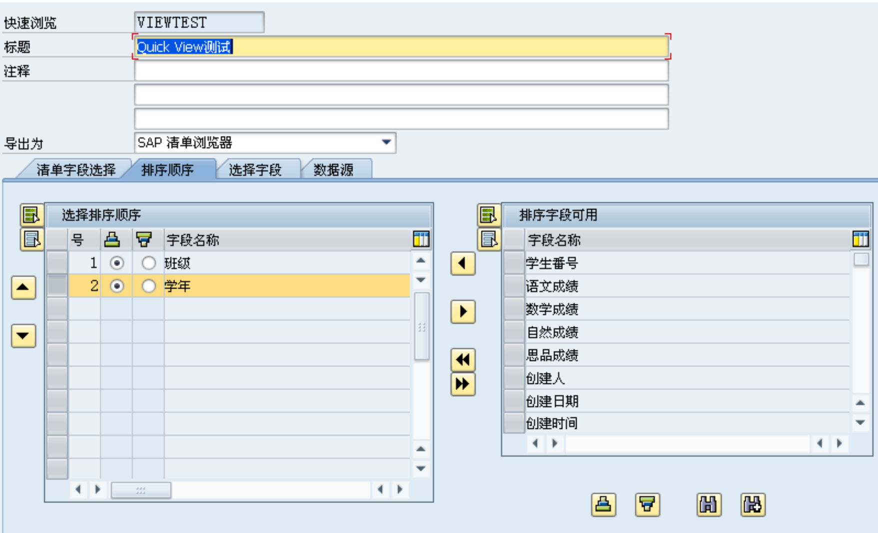


图 6–4

下面是“Selection Fields”视图，在这里指定选择屏幕上将出现的字段，如图 6–5 所示。

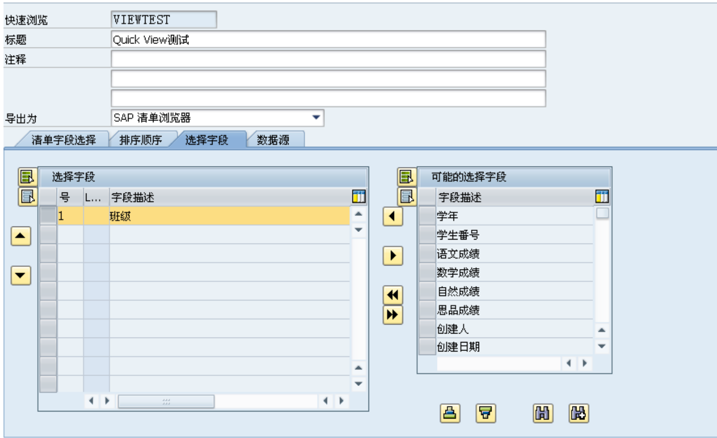


图 6–5

步骤四：Quick View（保存编辑视图）。

单击标准工具栏中的保存按钮，如图 6–6 所示。



图 6–6

2．Quick View的使用

步骤五：Quick View（运行视图）。

保存视图并单击执行按钮，测试运行该列表，屏幕如图 6–7 所示。



图 6–7

单击按钮显示视图，如图 6–8 所示。

Quick View测试

Quick View测试

班级	学年	学生番号	语文成绩	数学成绩	自然成绩	思品成绩
1A	2009	00001	60	68	70	81
		00002	62	65	74	83
		00003	62	65	71	89
		00004	61	63	75	88
		00005	64	66	75	88
		00006	64	66	75	88
		00007	64	66	76	88
		00008	62	66	72	88
		00009	62	63	72	82
		00010	62	63	74	85
			623	651	734	860

图 6–8

注：在 QuickView 初始画面输入 QuickView 名称，单击菜单按照以下路径可以查找报表名称。这样能在 ABAP 编辑器中查看系统自动生成的代码，也能够为该列表分配一个事物代码，如图 6-9 和图 6-10 所示。

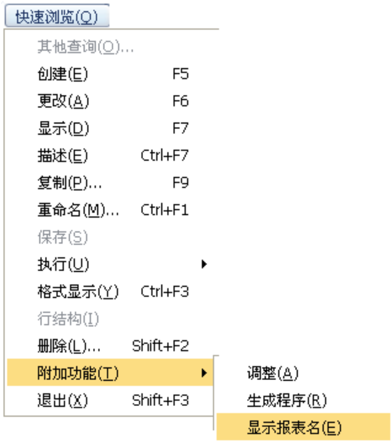


图 6-9



图 6-10

因为 SAP 没有将 QuickView 作为传输系统中的一个组件（即创建的 QuickView 不会出现在 Object Directory 中），故 QuickView 不能传输。另外，QuickViews 定义成属于某一系统用户，A 用户定义的 QuickView，B 用户无法使用。

6.1.2 Quick Query

可以将 Quick Query 理解成 QuickView 的高级版本，它在 QuickView 的基础上增加的功能有：

- 能够实现统计、排序、计算等功能。
- 提供标准的图形形式显示列表。
- 用户之间能够共享。
- 作为 SAP 传输系统的一个部分，能够在系统之间传输。

在创建 SAP Query 之前，先要弄明白两个概念：

- 功能区：在这里定义查询数据源，也称为信息集。
- 用户组：为了方便管理，定义一些用户组，将用户包含在用户组中。然后将功能区分配给用户组使用。用户组与用户、用户组与功能区都是多对多的关系。

1. 创建功能区

功能区访问路径：

菜单路径：工具→ABAP 工作台→实用程序→SAP 查询→信息集。

事务代码：SQ02。

步骤一：功能区（第一画面）。

在相应的栏位填入信息集的名称，单击 创建 按钮，如图 6-11 所示。



图 6-11

输入要创建的功能区名称，单击“创建”按钮，弹出对话框如图 6-12 所示。

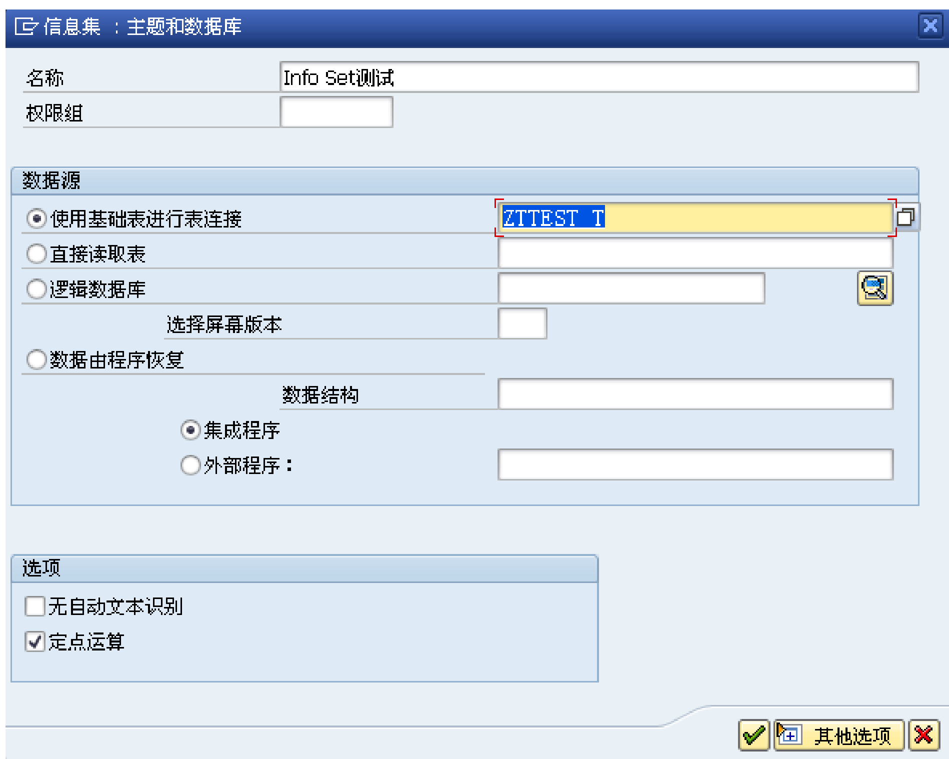


图 6-12

输入 InfoSet 的名称和基础数据表，单击✔按钮，进入数据源设定界面，如图 6-13 所示。

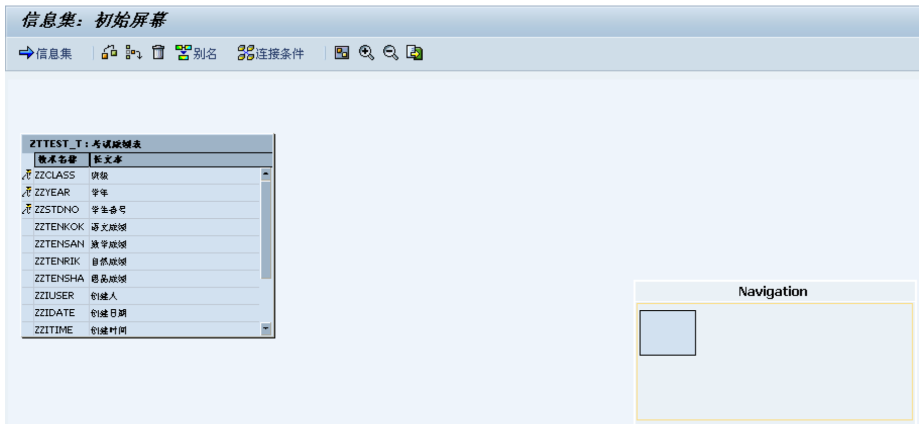


图 6–13

步骤二：功能区（第二画面）。

单击应用工具栏中的按钮，添加与之关联的表或视图，如图 6–14 所示。

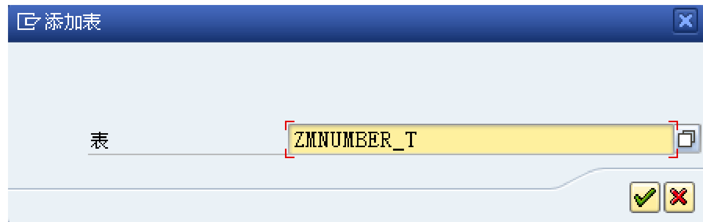


图 6–14

填写要追加的表，单击按钮，如图 6–15 所示。

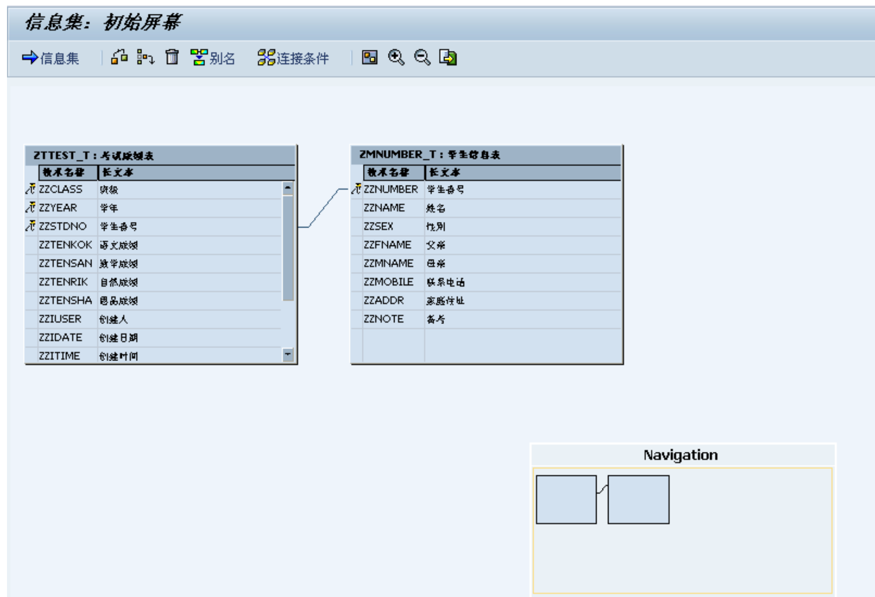


图 6–15

步骤三：功能区（第二画面）。

单击应用工具栏中的 ➡信息集 按钮，编辑输出信息集，如图 6–16 所示。

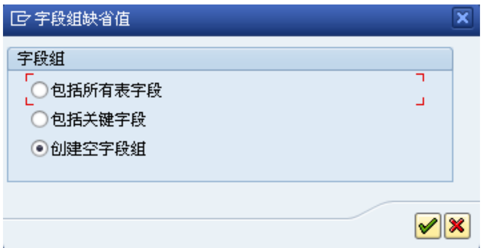


图 6–16

选中“创建空字段组”，单击 ✔ 按钮，进入如图 6–17 所示界面。

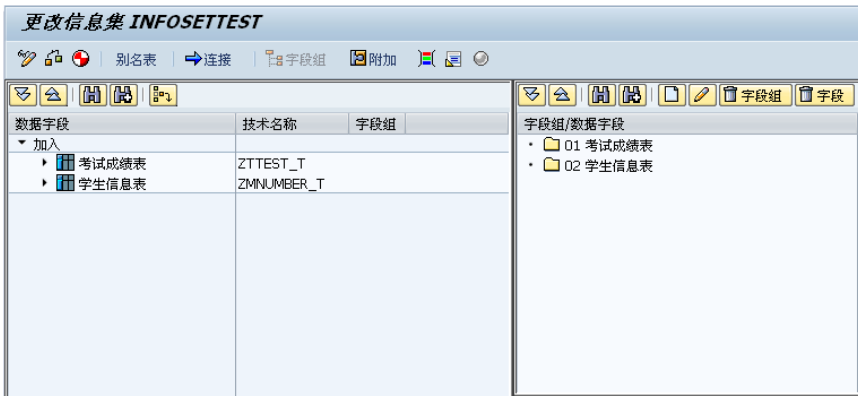


图 6–17

从左边的数据库表中选择字段，拖到右边的“Field Group Fields”中。在右下方，能够设定字段的显示标题，如图 6–18 所示。

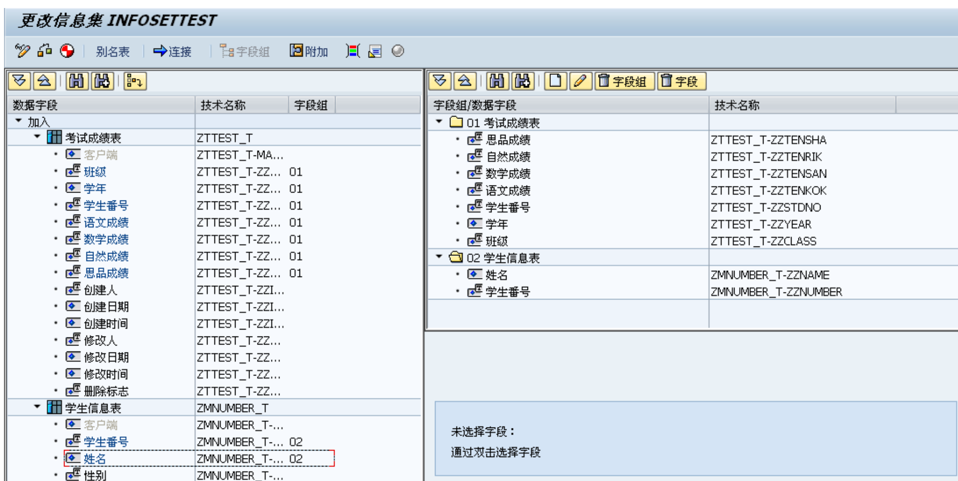


图 6–18

步骤四：QuickQuery（第二画面，保存、检查并生成该数据源）。

将需要的字段添加到字段中后，单击标准工具栏上的保存按钮，如图 6–19 所示。

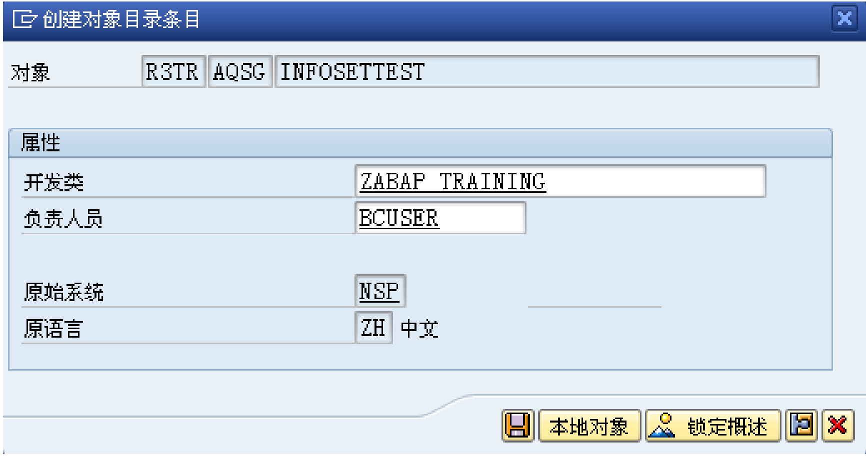


图 6–19

单击保存按钮，如图 6–20 所示。



图 6–20

单击上图中的按钮，如图 6–21 所示。



图 6–21

单击标准工具栏中的检查按钮，如图 6–22 所示。



图 6–22

单击应用工具栏中的生成按钮，如图 6–23 所示。



图 6–23

2．创建用户组

用户组访问路径：

菜单路径：工具→ABAP 工作台→实用程序→SAP 查询→用户组。

事务代码：SQ03。

步骤一：用户组（第一画面）。

在相应的栏位填入用户组的名称，单击 创建 按钮，如图 6–24 所示。



图 6–24

弹出创建对话框，如图 6–25 所示。



图 6–25

输入用户组的描述后，单击 保存 按钮，如图 6–26 所示。

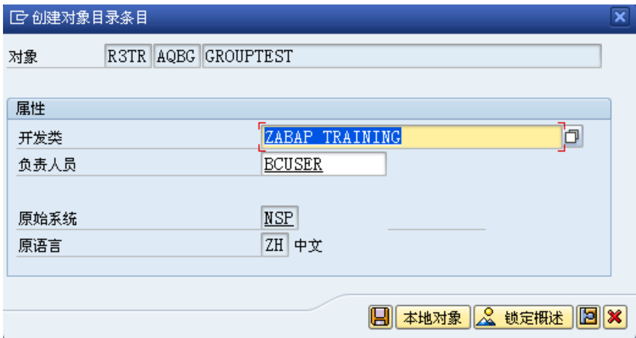


图 6–26

单击保存按钮，如图 6–27 和图 6–28 所示。

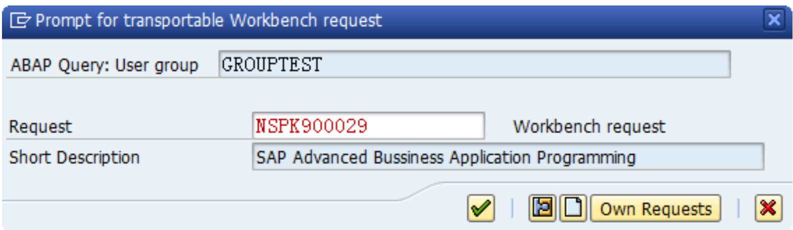


图 6–27



图 6–28

单击上图中的按钮。

步骤二：用户组（第二画面）。

单击 分配用户和功能区 按钮，填写可分配用户如图 6–29 所示。



图 6–29

单击分配信息集按钮，选中上述步骤创建的信息集，如图 6–30 所示。



图 6–30

步骤三：用户组（第二画面）。

单击标准工具栏中的保存按钮，保存用户组，如图 6–31 所示。



图 6–31

3．创建Query

用户组访问路径：

菜单路径：工具→ABAP 工作台→实用程序→SAP 查询→查询。

事务代码：SQ01。

步骤一：查询（第一画面）。

在相应的栏位填入查询的名称，单击 创建 按钮，如图 6–32 所示。



图 6–32

弹出如图 6-33 所示窗口，选择相应数据源。

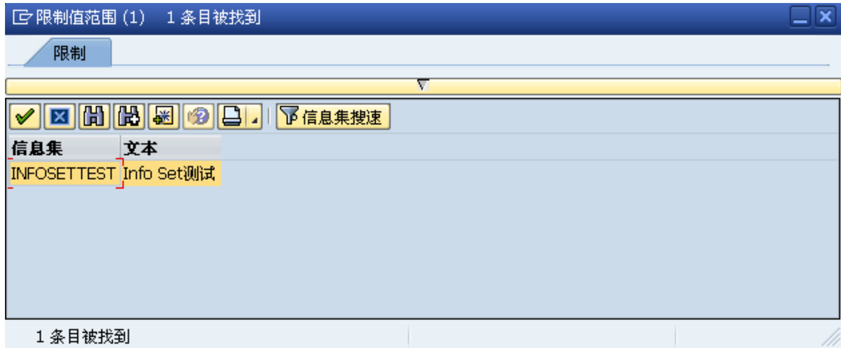


图 6-33

步骤二：查询（第二画面）。

输入标题，选择清单格式、输出格式等列表基本信息，如图 6-34 所示。



图 6-34

单击按钮，进入下一屏幕，如图 6-35 所示。



图 6-35

选择相应字段组，单击按钮，进入下一屏幕，如图 6-36 所示。

图 6-36

选择希望出现在选择屏幕及清单上的字段，单击按钮，进入下一屏幕，如图 6-37 所示。

图 6-37

在这个屏幕中，各选项的介绍如下：

- 勿使用参数 ID 预分配选项：选择字段是否自动带出默认值。
- 否：选择参数在选择屏幕上的输出次序。
- SV：如果选中该选项，则该选择参数只有一个输入框。
- 1Z：如果选中该选项，则该选择参数没有“多项选择”按钮。

步骤三：查询（第二画面）。

单击基本清单按钮，定义查询基本列表，选中并编辑要显示的字段，屏幕如图 6-38 所示。

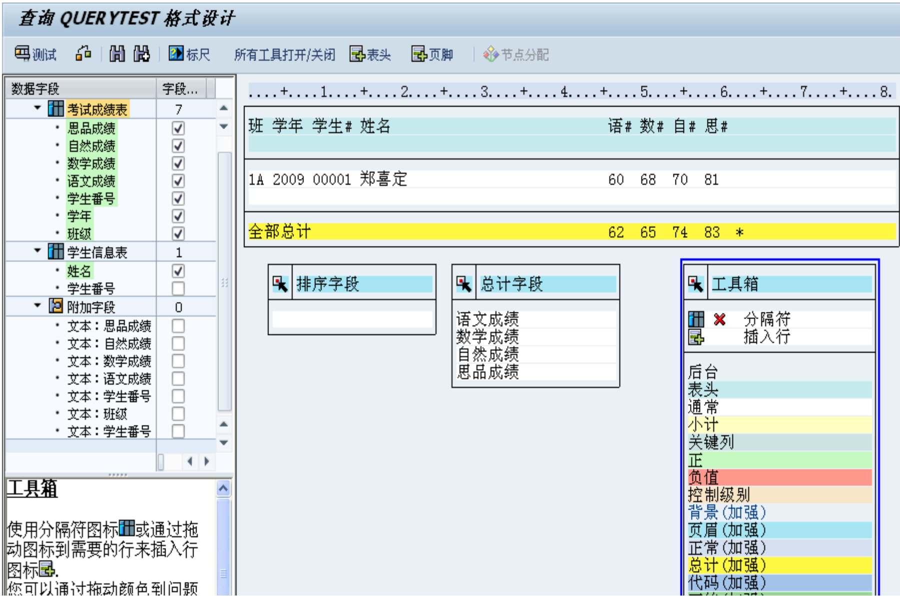


图 6-38

逐一双击需要出现在列表中的字段，设计字段清单的列。然后选中需要设定格式的列表字段，在左下角修改字段的显示格式，如图 6-39 所示。

清单字段:

班级

技术名称: ZTTEST_T-ZZCLASS

输出长度 (标准 2)　2

输出位置　1

清单字段颜色

行颜色

☐ 如果 <> 0 则仅显示字段

Apply　Reset

图 6-39

单击标准工具栏中的保存按钮，保存基本清单，单击按钮返回查询处理界面。

步骤四：查询（第二画面）。

单击统计按钮，定义查询统计功能，选中并编辑字段的统计选项，屏幕如图 6-40 所示。

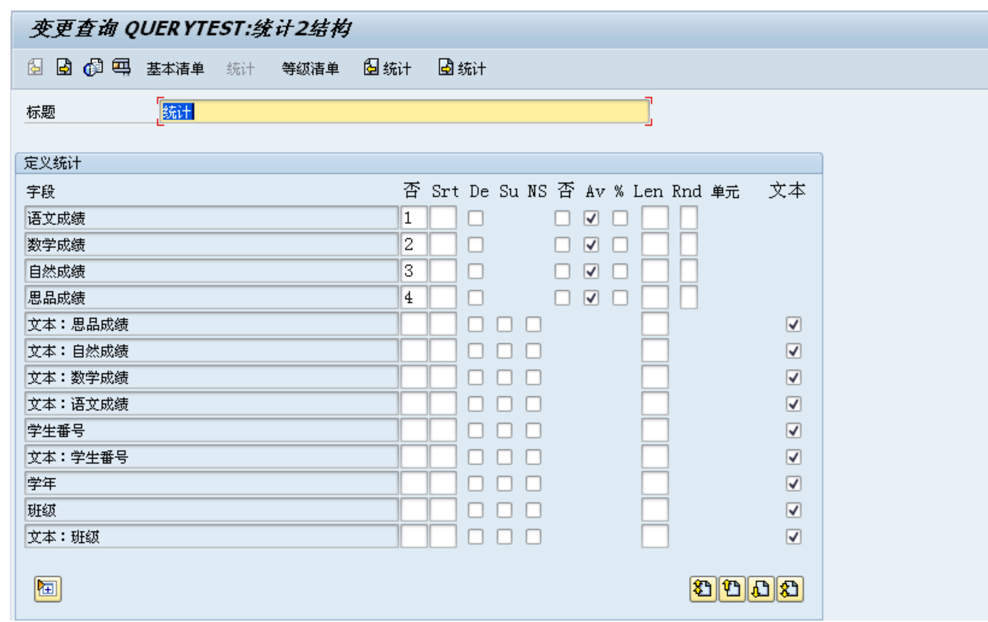


图 6-40

在这个屏幕中，各选项的介绍如下：

- 否：统计中字段顺序的定义，可以分配 1 和 98 之间的序列号。
- Srt：策略里的排序顺序的定义。
- De：如果选择此字段，统计将根据相关排序标准以降序排序，在其他情况中，以升序排列。
- Su：ABAP/4 查询在统计中为排序标准生成小计，输出含相关小计的行。
- NS：在统计中为每个字符串插入换页，即每次排序字符串更改，就显示一个新的页面。
- 否：盘点用于确定统计行的数据记录。
- Av：为左侧单个被加数输入的字段计算平均值。
- %：输出总计字段百分比。
- Len：更改字段的输出长度。
- Rnd：整取时的小数点的位数。

使用统计列表，有三个限制条件：

- 它的字段来源为基本列表。
- 只能对数量和金额字段进行条目数统计、平均值统计、百分比统计。
- 显示统计列表时，如果与“数量/金额”相应的“单位/币别”不一致，则能引发统计错误。

单击标准工具栏中的保存按钮，保存统计设定，单击按钮返回查询处理界面。

步骤五：查询（第二画面）

单击等级清单按钮，定义查询等级功能，选中并编辑字段的等级选项，屏幕如图 6-41 所示。



图 6-41

在这个屏幕中，各选项的介绍如下：

- 否：统计中，字段顺序的定义；可以分配 1 和 98 之间的序列号。
- Crit：左侧选择指定的字段作为优先级清单标准。优先级清单将根据优先级清单标准排序。
- Asc：按优先级列表对优先级列表进行排序或按升序排列。否则按降序排序。
- Len：更改字段的输出长度。
- Rnd：整取时的小数点的位数。

单击标准工具栏上的保存按钮，保存等级设定，单击按钮返回查询处理界面。

步骤六：查询（第二画面）。

单击输出次序按钮，定义查询输出次序，编辑输出序列，屏幕如图 6-42 所示。



图 6-42

单击复制按钮。

步骤七：查询（第二画面）。

单击标准工具栏中的保存按钮，如图 6-43 所示。

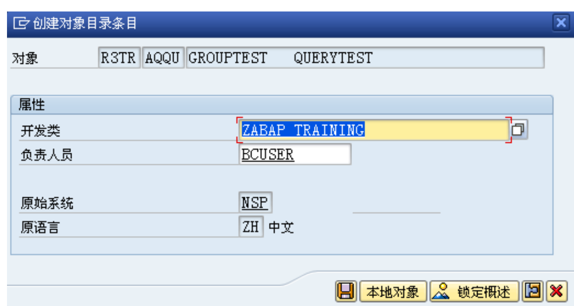


图 6-43

单击上图中按钮。

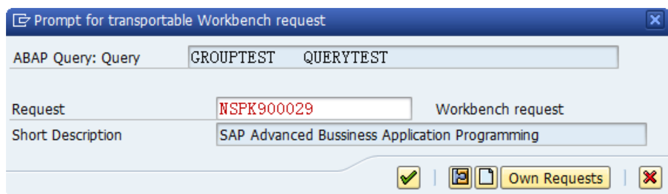


图 6-44

单击上图中的按钮，如图 6-45 所示。



图 6-45

4．运行查询

单击应用工具栏中的按钮，运行结果如图 6-46 所示。



图 6-46

单击图 6-46 中的按钮。

■ 选择屏幕如图 6-47 所示。

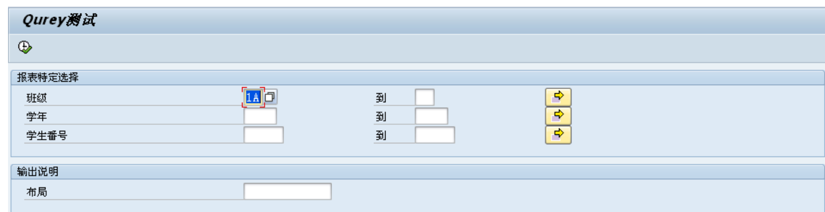


图 6-47

■ 基本列表如图 6-48 所示

Qurey测试

Qurey测试

班	学年	学生番号	姓名	Σ语文成绩	Σ数学成绩	Σ自然成绩	Σ思品成绩
1A	2009	00001	郑喜定	60	68	70	81
1A	2009	00002	魏喜奎	62	65	74	83
1A	2009	00003	金喜善	62	65	71	89
1A	2009	00004	洪喜官	61	63	75	88
1A	2009	00005	朴喜珍	64	66	75	88
1A	2009	00006	王喜凤	64	66	75	88
1A	2009	00007	刘喜仁	64	66	76	88
1A	2009	00008	白喜医	62	66	72	88
1A	2009	00009	赵喜庆	62	63	72	82
1A	2009	00010	张喜早	62	63	74	85
				623	651	734	860

图 6-48

6.2 SAP快速录入工具

SAP 数据导入方式从其版本升级、对数据精度的要求以及数据的实时性，可分为以下几种类型导入工具：

■ 直接导入（Direct Input）

为了增强 Batch Input 的处理，系统提供了 Direct Input 技术，特别是用在大批量的数据传输处理上。和 Batch Input 相比，Direct Input 并不创建 session，也不处理屏幕（screen），而是直接存储数据。为了直接存储数据到相应的数据库表中，系统调用许多函数模块（Function Module）来对数据进行必要的检查。在数据的传输过程中，如果发生错误，Direct Input 技术提供了一种重新开始（Restart）的机制。然而，为了激活这种机制，Direct Input 程序只能在后台执行。

■ 扩展计算机辅助测试工具 CATT 和 ECATT（Extended Computer Aided Test Tool）

SAP 系统有一个庞大的数据库，对大量的数据操作肯定是用批量处理的方法。除了系统本身提供的程序外，SAP 还允许用户使用 ABAP 自行开发应用程序，以满足实际业务的需求。实际上，需要批量处理数据的业务种类很多，不可能为所有的业务流程都开发应用程序，一是程序开发周期较长，影响业务进度；二是工作量巨大，需要配备大量的程序员。另外考虑到有些业务流程只是短期需求，程序利用率不高，造成资源浪费。为此，SAP 提供了一种称为 CATT 的批处理模式，它在前台表现为一个录屏的过程，利用系统中原有的交易代码或操作命令进行一系列操作的组合，制作一个用户自定义的操作模块，根据用户的变量设置，实现函数化的数据操作。CATT 的优点在于开发周期短、操作简单、修改灵活，可以实现模块共享，也可以实现跨 Client 的传递。

在 SAP 系统中，CATT 是批量导入各类企业原始数据的一个非常实用的工具。它的功能是先让使用者操作一遍，让系统记录下使用者的操作，做适当的调整后，导出一个输入格式文件，将要输入系统的数据按格式输入到文件中，再在 CATT 中用此文件沿用之前录制的操作让 SAP 系统逐个完成录入工作。

ECATT 是 SAP 自带的功能测试工具，主要目标用来做 SAP 业务流程的自动化测试，每个测试都会生成一个详细的日志来显示测试流程和测试结果。和 CATT 相比，CATT 操作简单，类似于 BDC 的录屏功能，用于常用的简单业务数据导入，基本已被 ECATT 完全取代。除了 CATT 功能之外，ECATT 主要应用于基于 Gui Windows/Gui Java 自动化测试，同时也为外部工具提供了第三方接口（获 BC-ECATT 授权）。所有的测试脚本，无论是 ECATT 自己生成的还是外部工具的，都应该具有相同的数据对象类型，与 SAP 结合在一起并保存在相同的位置。在 R/3 的功能测试中，ECATT 录屏功能简单实用，测试脚本、测试数据、系统数据分别存放，可重复使用，并附有详细测试报告以及纠错功能，作为自带工具简单实用。对于 Web 方面的测试的话就要借用第三方的测试软件，如知名的 Mercury 的 QTP。

在新版本的 R/3 中 CATT 已经被废弃了，本书不做过多介绍。ECATT 是基于录屏，但更偏重于测试，提供了第三方测试的接口。由于 NetWeaver 的推广，同时它也支持 Web 的功能测试（不支持 Web Dynpro）；测试脚本、测试数据、系统数据分开维护，利于复用；基于自动化测试设计，可以节约大量测试时间；支持 CATT 到 ECATT 的迁移。在 R/3 的功能测试中，ECATT 优势很明显，录屏功能简单实用，测试脚本、测试数据、系统数据分开置放，有利于重复使用，还有相应的测试报告和错误纠正，简单而高效。唯一的不足就是，由于不支持 Web Dynpro 的测试，所以关于这类测试还是要使用到第三方的测试软件，具体操作本书不做过多介绍。

■ 旧系统迁移工作台 LSMW（Legacy System Migration Workbench）

旧系统迁移工作台是 SAP 用来从旧的系统或非 SAP 系统把大批量或周期性的数据迁移到 SAP 的主力工具，类似于 ECATT/SCATT，其优点可以在一个 Session 中处理大批量数据，并灵活运用。如果用 BDC 和 ECATT 只能根据定制好的画面输入参数。一旦没有这个值系统就会报错。比如根据业务对 FI Account Document 或者主数据一些参数作了扩展，一旦参数缺失或者不匹配，不得不终止 Session，通过 SM35 手动去更正或者重新录屏。LSMW 具有一般的通用性，它导入模板的设计过程包括 15 个步骤，每个都是独立的单元并可进行单独的更新和修改，而不影响其他模块，只需做出相应 mapping，将必要的数据指定对应的字段就可以了，无须额外的编码。

当然，LSMW 的局限性在于只能导入固定类型的主数据，如 Customer/Vendor Master、Material Master、Financial Document、G/L Account。如果是自定义的 ABAP 程序来导入主数据就行不通了。同样做了一个 FI Account Document 的例子，对出现重复统计的 Line Item 也显得不是特别得心应手。

在 SAP 系统中，批处理操作有多种方法。如果是对一个事物码（T-CODE）进行批处理操作，常用的是 LSMW。LSMW 全称是 Legacy System Migration Workbench，它能够对静态数据（如各个主数据）、动态数据（如初始化库存）、业务数据（如销售订单）等进行批量操作，是上线数据准备的主力武器。

LSMW 的原理是对需要批处理操作的流程进行录像，然后设定模板，再将准备好的数据传入到 SAP 系统进行预转换，如果合适就进行实际的转换。

LSMW 导入数据操作分为以下几大步骤（见图 6−49）：

■ 准备需要导入的数据。

■ 转换模板定义。

■ 读取数据并预转换。
■ 实际转换。

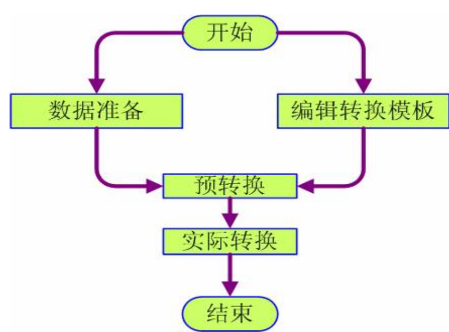


图 6-49

在操作界面有 20 个步骤，而在 ECC 6.0 中，标准步骤只有 14 个，按 14 个步骤进行讲解。

1．操作界面说明

步骤一：登录 LSMW。

批导入的事物码（TCODE），在主窗口界面输入就可以进入，如图 6-50 所示。

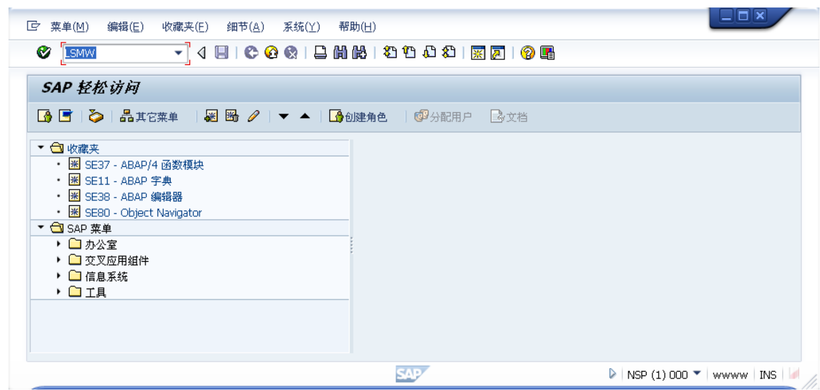


图 6-50

输入事物码后，如果是第一次运行，则出现如图 6-51 所示界面。

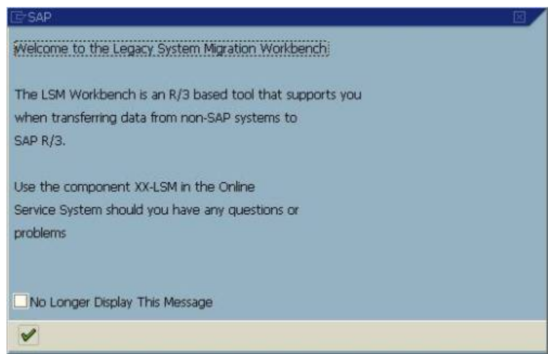


图 6-51

这是一个欢迎界面，按确认键进入到 LSMW 的管理界面，如图 6–52 所示。

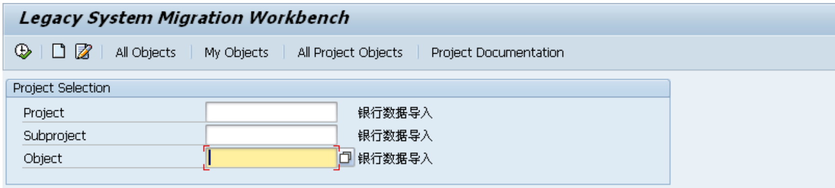


图 6–52

这个界面主要解决以下几方面问题：

（1）批导入对象的管理，包括新增、修改、删除、查找、导出、导入等操作。

（2）进入到其他工作界面，主要有录像操作界面、分步操作界面。

步骤二：登录工程，子工程及对象。

如果要进行一个批处理操作，则需要输入或通过选择确定 Project、Subproject、Object，如图 6–53 所示，然后再进行具体的操作。

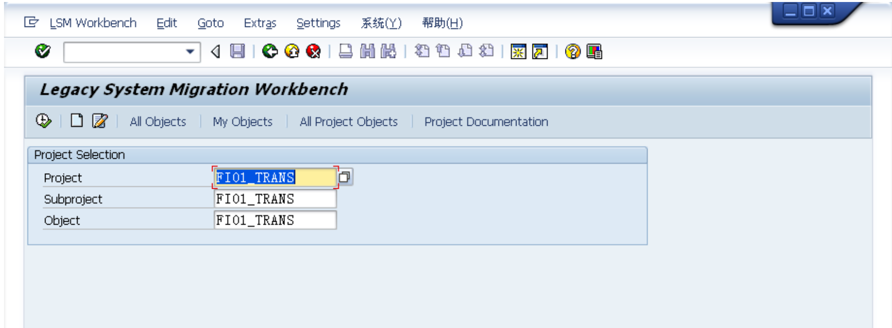


图 6–53

单击按钮，如图 6–54 所示。

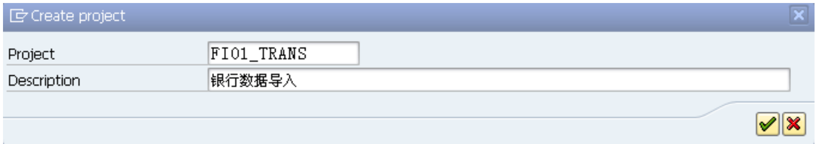


图 6–54

填入描述，按回车键或单击按钮，如图 6–55 所示。

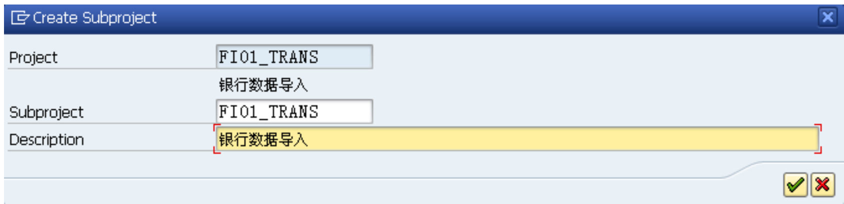


图 6–55

填入描述，按回车键或单击按钮，如图 6-56 所示。



图 6-56

填入描述，按回车键或单击按钮，结果如图 6-57 所示。

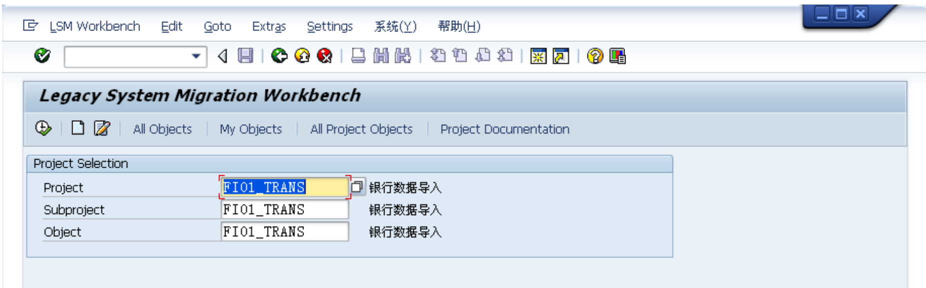


图 6-57

2. 分步操作界面

在管理界面选择 Project、Subproject、Object，按执行按钮进入分步操作界面，如图 6-58 所示。

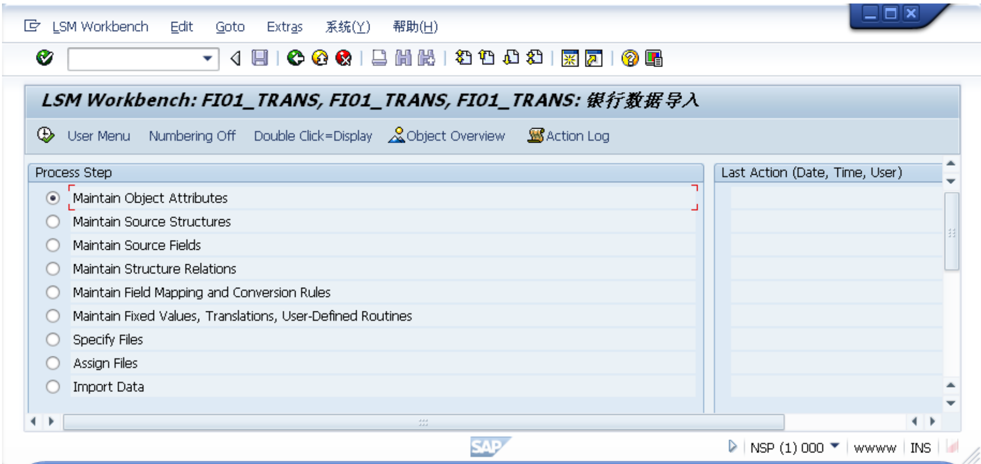


图 6-58

步骤一：维护对象属性。

选择步骤 Maintain Object Attributes，单击按钮，如图 6-59 所示。

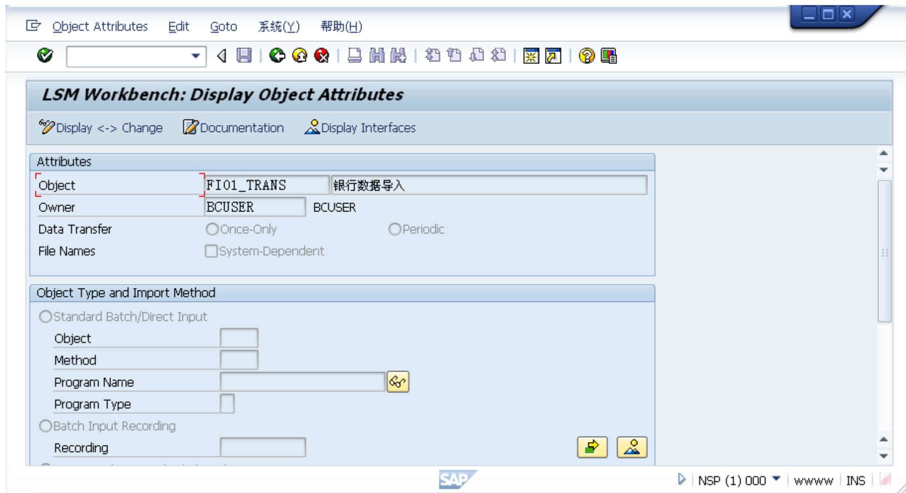


图 6-59

单击Display <-> Change按钮，转换到变更状态，如图 6-60 所示。

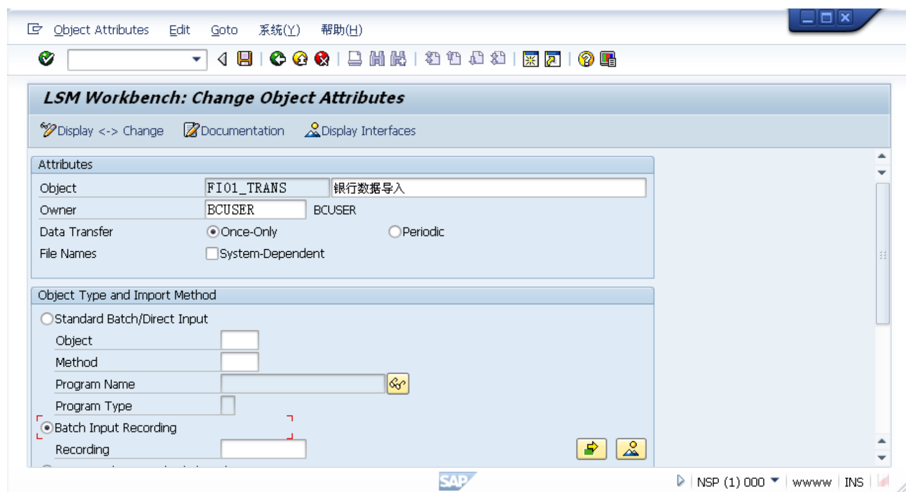


图 6-60

选中选项 Batch Input Recording 并单击按钮，如图 6-61 所示。



图 6-61

单击□按钮，新建批处理记录，如图 6–62 所示。

图 6–62

输入记录名称及描述，单击✔按钮，如图 6–63 所示。

图 6–63

输入事务代码 FI01 并单击✔按钮，如图 6–64 所示。

图 6–64

填写银行国家及银行代码，如图 6–65 所示。

图 6–65

填写银行名称、区域、城市、街道及分行信息，单击按钮，如图 6-66 所示。



图 6-66

单击 Default All 按钮，如图 6-67 所示。



图 6-67

注：利用 Default Reset Screen Field Screen Field Repeat Recording 功能编辑记录。

单击按钮，单击按钮返回，如图 6-68 所示。

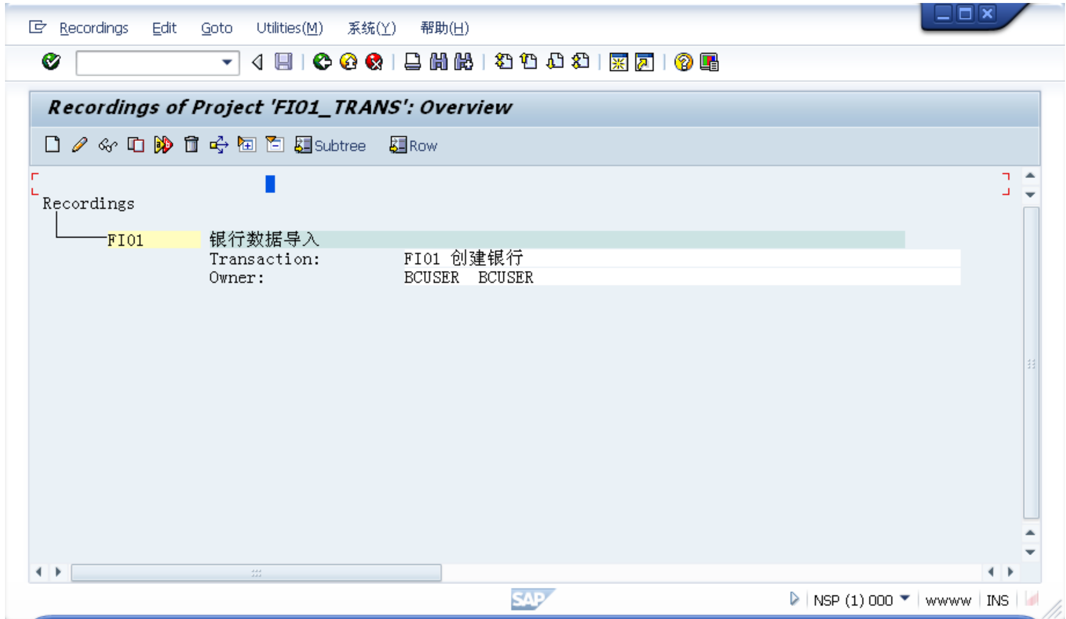


图 6-68

单击按钮，单击按钮返回主界面，如图 6-69 所示。

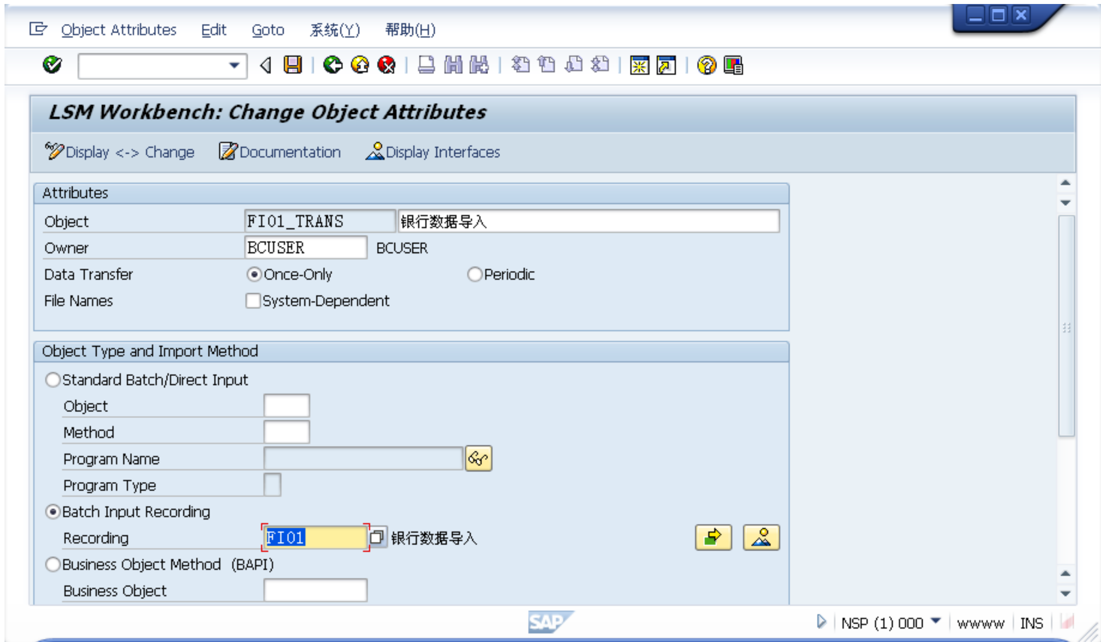


图 6-69

填入所建记录，单击按钮，单击按钮返回 LSMW 处理界面，如图 6-70 所示。

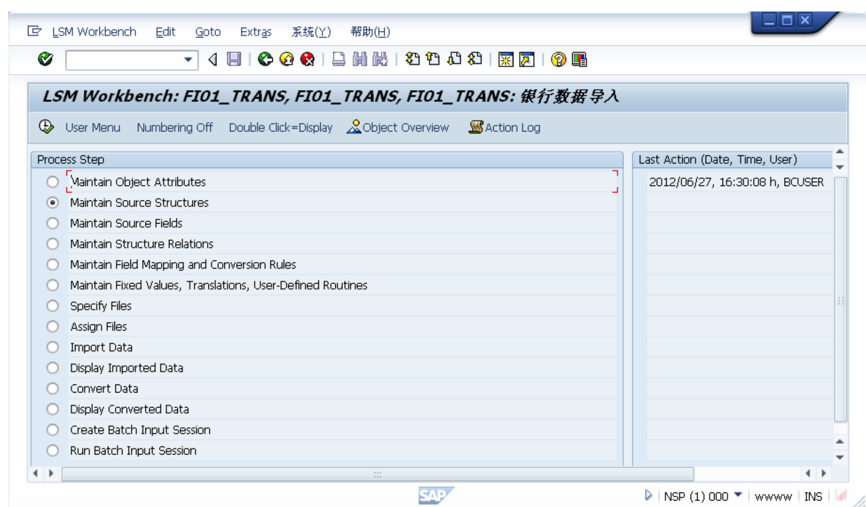


图 6-70

单击 Object Overview 按钮，显示记录信息，如图 6-71 所示。

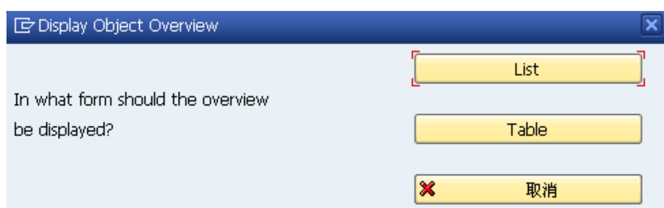


图 6-71

单击 Table 按钮，如图 6-72 所示。

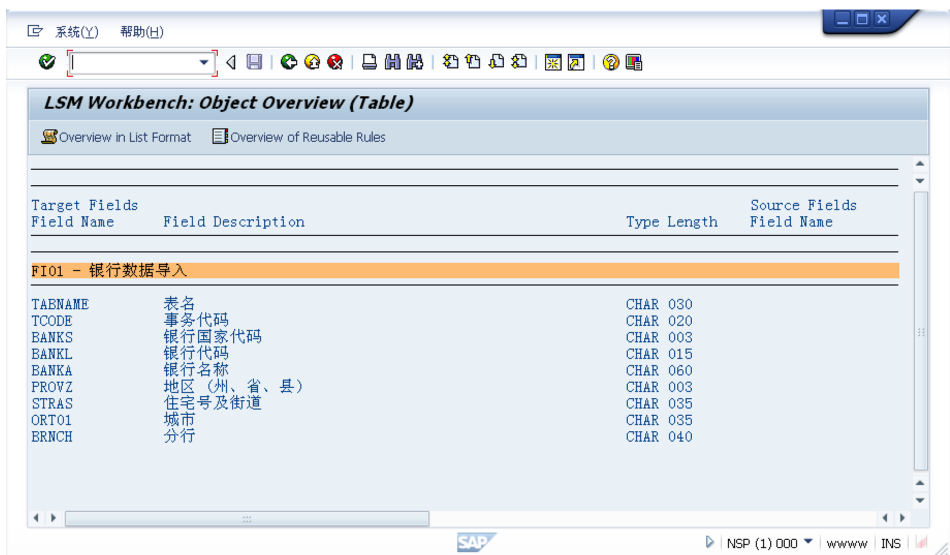


图 6-72

将文本粘贴至 Excel 文档，使用文本导入工具编辑文本，菜单如图 6–73 所示。编辑信息如图 6–74 所示。

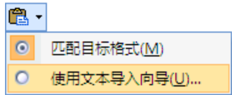


图 6–73

TABNAME	表名	CHAR	30
TCODE	事务代码	CHAR	20
BANKS	银行国家代码	CHAR	3
BANKL	银行代码	CHAR	15
BANKA	银行名称	CHAR	60
PROVZ	地区（州、省、县）	CHAR	3
STRAS	住宅号及街道	CHAR	35
ORT01	城市	CHAR	35
BRNCH	分行	CHAR	40

图 6–74

利用 Excel 查找和替换工具，将 CHAR 替换为 C，如图 6–75 所示。



图 6–75

结果如图 6–76 所示。

TABNAME	表名	C	30
TCODE	事务代码	C	20
BANKS	银行国家代码	C	3
BANKL	银行代码	C	15
BANKA	银行名称	C	60
PROVZ	地区（州、省、县）	C	3
STRAS	住宅号及街道	C	35
ORT01	城市	C	35
BRNCH	分行	C	40

图 6–76

步骤二：定义数据结构。

选择选项 Maintain Source Structures，单击按钮，如图 6–77 所示。



图 6–77

单击按钮，使当前状态为可编辑，如图 6-78 所示。



图 6-78

单击按钮，创建数据结构，如图 6-79 所示。

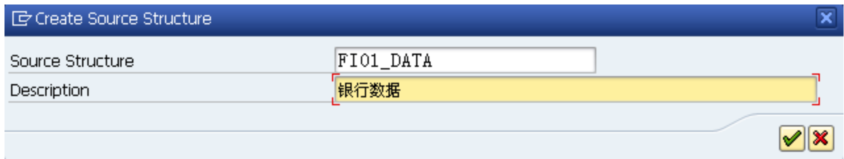


图 6-79

单击按钮，如图 6-80 所示。

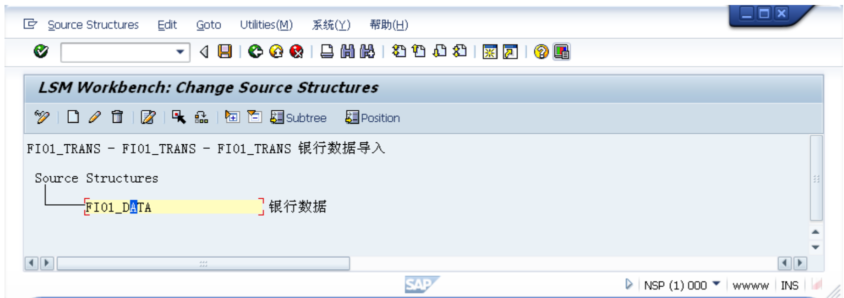


图 6-80

单击按钮，单击按钮返回 LSMW 处理界面。

步骤三：为结构添加字段

选择选项 Maintain Source Fields，单击按钮，如图 6-81 所示。



图 6-81

单击按钮，使当前状态为可编辑，如图 6-82 所示。

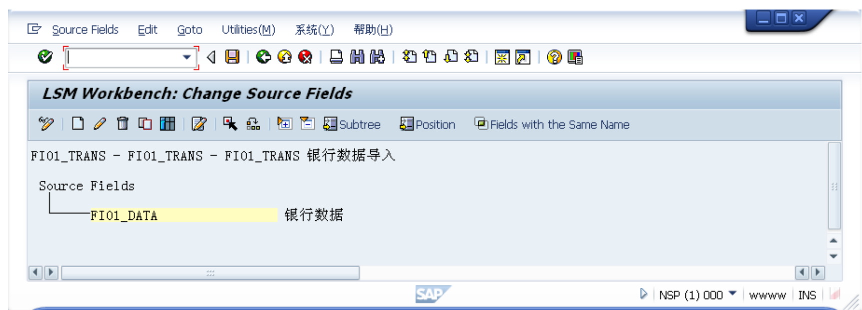


图 6-82

选中结构 FI01_DATA 并单击按钮，为结构添加字段，如图 6-83 所示。

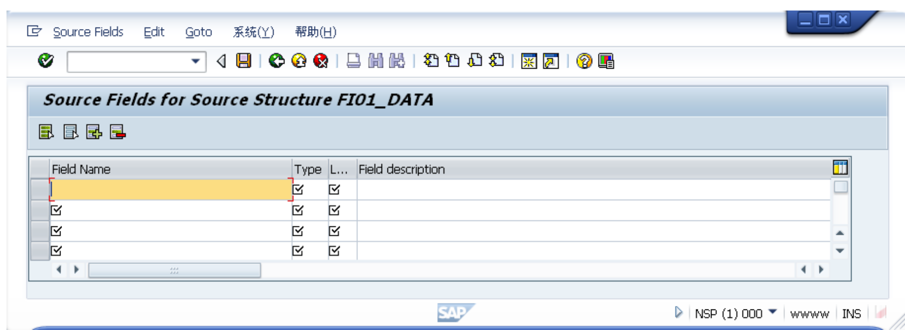


图 6-83

将步骤一中编辑的字段拷贝至表格中（省略前 2 项），编辑如图 6-85 所示。

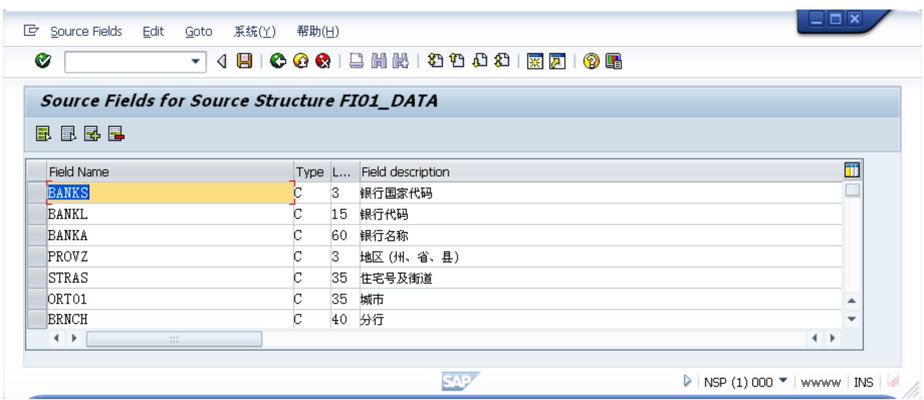


图 6-84

上图显示了源表结构输入界面，共有 4 列，需要分别填写：

■ 字段名（Field Name）：输入源表中的字段名，详见上面保存的 Excel 表中的表头。

■ 类型（Type）：数据类型，C 为字符型。

■ 长度（Length）：字段长度。

■ 描述（Field Description）：字段描述，可选项。

单击💾按钮，单击⬅按钮返回主处理界面，如图 6-85 所示。

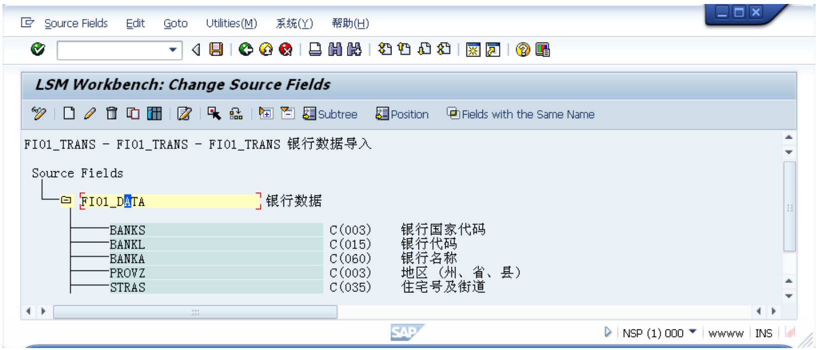


图 6-85

单击💾按钮，单击⬅按钮返回 LSMW 处理界面，如图 6-86 所示。

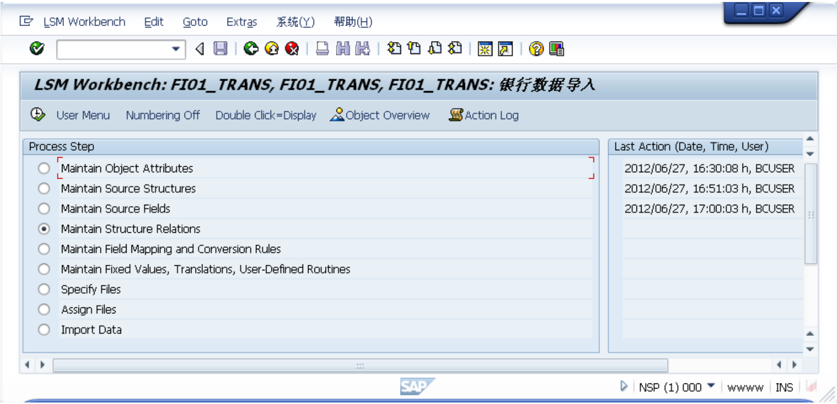


图 6-86

步骤四：创建数据结构与导入对象间的对应关系。

选中选项 Maintain Structure Relations，单击按钮，如图 6-87 所示。



图 6-87

单击按钮，使当前状态为可编辑，如图 6-88 所示。

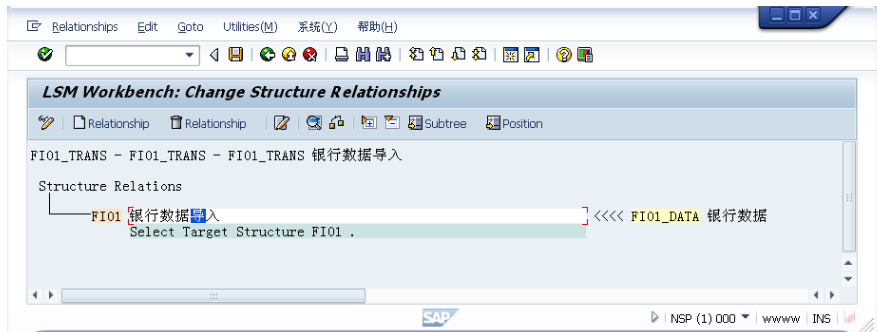


图 6-88

单击按钮，单击按钮返回 LSMW 处理界面，如图 6-89 所示。



图 6-89

步骤五：创建数据结构与导入对象间字段的映射关系。

选中选项 Maintain Field Mapping and Conversion Rules，单击按钮，如图 6-90 所示。



图 6-90

单击按钮，使当前状态为可编辑，如图 6-91 所示。

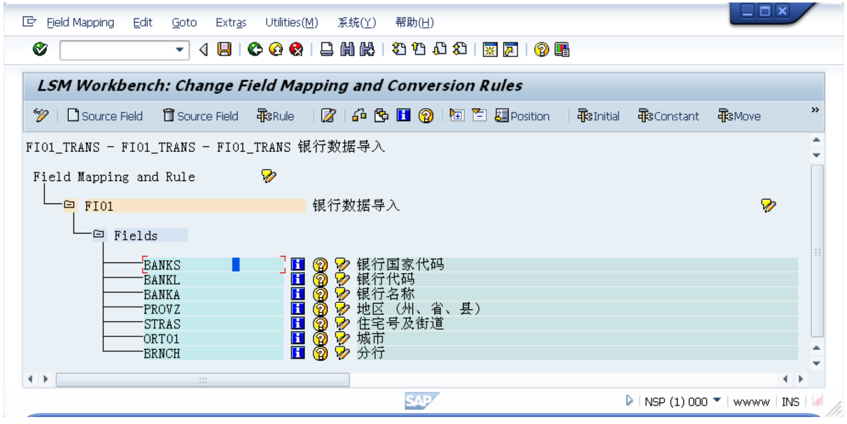


图 6-91

选中字段 BANKS 并单击 Source Field 按钮，如图 6-92 所示。

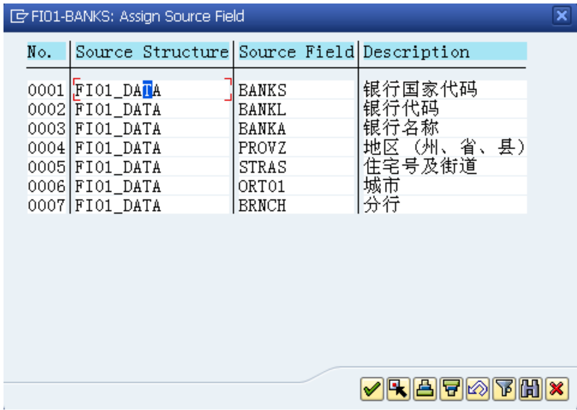


图 6-92

选中银行国家代码并单击按钮，或者双击银行国家代码，如图 6-93 所示。

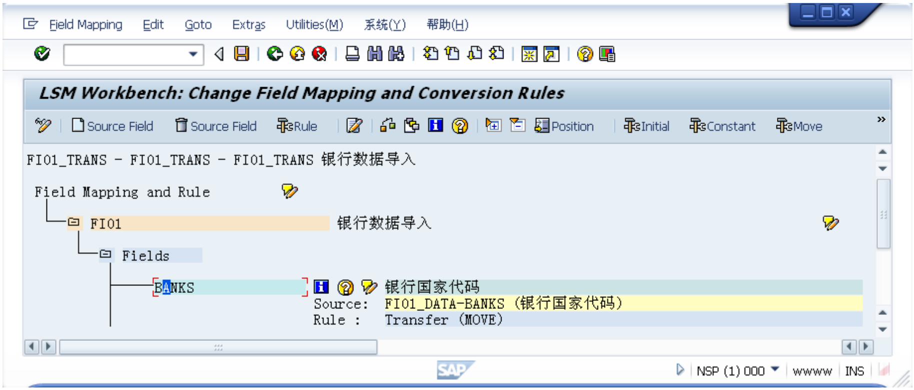


图 6-93

单击Rule按钮，可利用方法如图 6-94 所示。

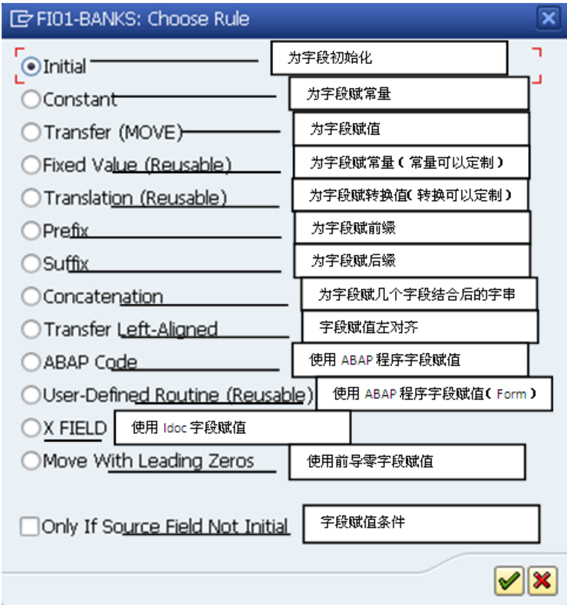


图 6-94

双击字段 BANKL 并单击 Source Field 按钮，重复以上步骤，编辑影射规则如图 6-95 所示。



图 6-95

单击按钮，单击按钮，检查无误后单击按钮返回 LSMW 处理界面，如图 6-96 所示。



图 6-96

此处用于定义字段影射过程中出现的常量，转换及用户自定义程序，如图 6-97 所示。

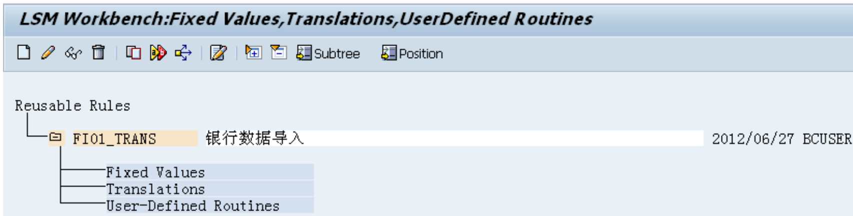


图 6-97

由于本例中并未涉及这里不做介绍。

步骤六：声明导入文件。

选中 Specify Files 选项，单击按钮，再单击按钮，使当前状态为可编辑，如图 6-98 所示。



图 6-98

选中 Legacy Data　On the PC （Frontend）选项，单击按钮，定义文件及文件格式如图 6-99 所示。

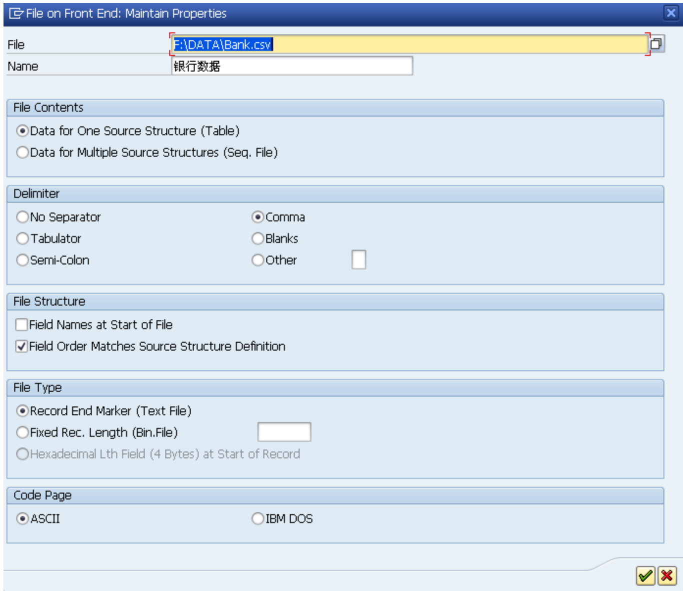


图 6-99

“File”项输入源表的文件名（一般为文本文件），“Name”项输入说明。“Delimiter”指定文件的分隔符，文件的分隔符是〈Tab〉键时应选中 Tabulator；本例中分隔符是逗号，所以选择“Comma”。“Field Name At Start Of File”项指定第一行是否有字段名，文本文件的第一行是字段名时需要选中。“Field Order Matches Source Structure Definition”项指定字段顺序是否与源表数据相同。

单击按钮，声明结果如图 6-100 所示。

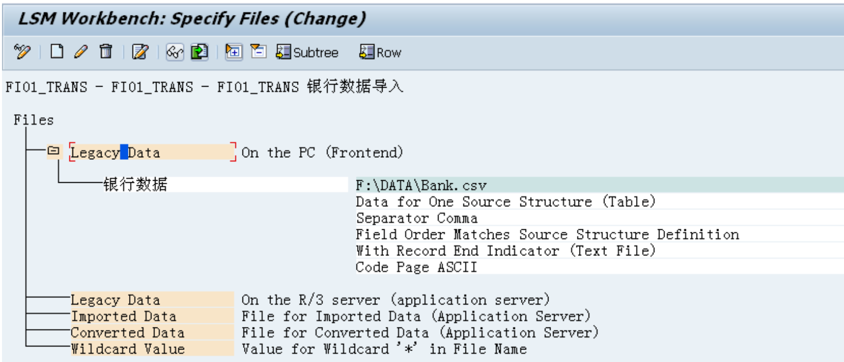


图 6-100

文件数据编辑如图 6-101 所示。

A	B	C	D	E	F	G	H	I
CN	10000051	中国银行	10	北京朝阳区鲁迅街	北京	朝阳分行		
CN	10000052	中国银行	10	北京海淀区长安街	北京	海淀分行		
CN	10000053	中国银行	10	北京四临区胡同街	北京	四临分行		
CN	10000054	中国银行	10	北京中山区步行街	北京	中山分行		
CN	10000055	中国银行	10	北京高新区火炬街	北京	高新分行		
CN	10000056	中国银行	10	北京沙河区爱仙街	北京	沙河分行		
CN	10000057	中国银行	10	北京千经区人仙街	北京	千经分行		
CN	10000058	中国银行	10	北京维和区鲁班街	北京	维和分行		
CN	10000059	中国银行	10	北京六口区眼光街	北京	六口分行		
CN	10000050	中国银行	10	北京朝阳区王府井街	北京	朝阳分行	E89032	AA

图 6-101

注：导入文件既可以作为前台 PC 机上的文件，也可以是后台 Server 上的文件，本例中为前台文件，关于后台文件作为导入文件本书不做阐述。

单击按钮，单击按钮返回 LSMW 处理界面。

步骤七：确定导入文件与导入结构之间的关系。

选中 Assign Files 选项，单击按钮，单击按钮，使当前状态为可编辑，如图 6-102 所示。

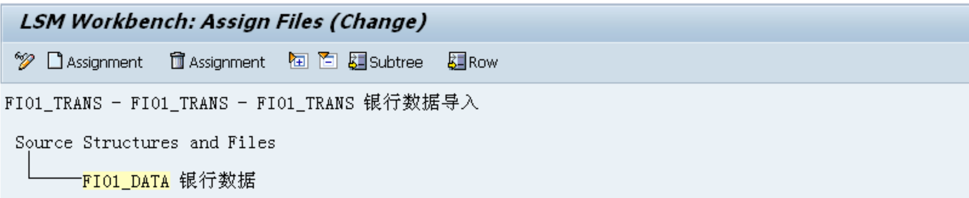


图 6-102

选中 FI01_DATA 银行数据，单击 Assignment 按钮，将文件影射原结构体，如图 6-103 所示。

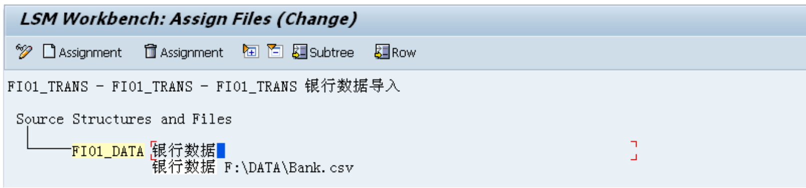


图 6-103

单击按钮，单击按钮返回 LSMW 处理界面。

步骤八：导入文件数据。

选中 Import Data 选项，单击按钮，显示如图 6-104 所示。

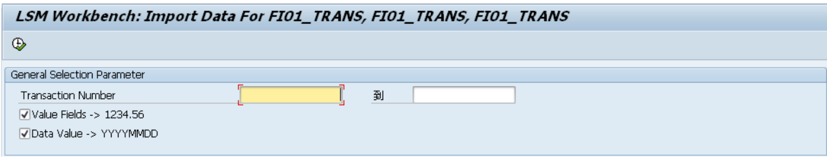


图 6-104

确定导入条件，单击按钮，如图 6-105 所示。

```
LSM Workbench: Import Data For FI01_TRANS, FI01_TRANS, FI01_TRANS

LSM Workbench: Import Data For FI01_TRANS, FI01_TRANS, FI01_TRANS
2012/12/04 - 14:36:55
File(s) Read:       F:\DATA\Bank.csv
File Written:       FI01_TRANS_FI01_TRANS_FI01_TRANS.lsmw.read
Source Structure             Read      Written       Not Written
FI01_DATA                      10        10                0
Transactions Read:                  10
Records Read:                       10
Transactions Written:               10
Records Written:                    10
```

图 6-105

单击按钮返回 LSMW 处理界面。

步骤九：确认导入文件数据。

选中 Display Imported Data 选项，单击按钮，显示如图 6-106 所示。



图 6-106

单击按钮，导入结果如图 6–107 所示。

LSM Workbench: Imported Data

Field Contents　Change Display　Display Colour Legend

File　FI01_TRANS_FI01_TRANS_FI01_TRANS.lsmw.read

Row	Structur	Conts.		
	LSMWFI01_TRANS　FI01_TRANS	FI01_TRANS　NSP　00020121204143648BCUSER		
1	FI01_DATA	CN 10000051	中国银行	010北京朝阳区鲁迅街
2	FI01_DATA	CN 10000052	中国银行	010北京海淀区长安街
3	FI01_DATA	CN 10000053	中国银行	010北京四临区胡同街
4	FI01_DATA	CN 10000054	中国银行	010北京中山区步行街
5	FI01_DATA	CN 10000055	中国银行	010北京高新区火炬街
6	FI01_DATA	CN 10000056	中国银行	010北京沙河区聚仙街
7	FI01_DATA	CN 10000057	中国银行	010北京千经区人仙街
8	FI01_DATA	CN 10000058	中国银行	010北京维和区鲁瑶街
9	FI01_DATA	CN 10000059	中国银行	010北京六口区眼光街
10	FI01_DATA	CN 10000050	中国银行	010北京朝阳区王府井街

图 6–107

单击按钮返回 LSMW 处理界面。

步骤十：转换确认导入文件数据。

选中 Convert Data 选项，单击按钮，显示如图 6–108 所示。

LSM Workbench: Convert Data For FI01_TRANS, FI01_TRANS, FI01_TRANS

General Selection Parameter

Transaction Number　到

图 6–108

单击按钮，转换结果显示如图 6–109 所示。

LSM Workbench: Convert Data For FI01_TRANS, FI01_TRANS, FI01_TRANS

LSM Workbench: Convert Data For FI01_TRANS, FI01_TRANS, FI01_TRANS

2012/12/04 - 14:44:10

File Read:　FI01_TRANS_FI01_TRANS_FI01_TRANS.lsmw.read
File Written:　FI01_TRANS_FI01_TRANS_FI01_TRANS.lsmw.conv

Transactions Read:　10
Records Read:　10
Transactions Written:　10
Records Written:　10

图 6–109

单击按钮返回 LSMW 处理界面。

步骤十一：确认转换后数据。

选中 Display Converted Data 选项，单击按钮，显示如图 6–110 所示。

Display Converted Data

Project	FI01_TRANS	银行数据导入
Subproject	FI01_TRANS	银行数据导入
Object	FI01_TRANS	银行数据导入
File	FI01_TRANS_FI01_TRANS_FI01_TRANS.lsmw.conv	
From Line		
To Line		

图 6–110

单击✔按钮，转换结果如图 6-111 所示。

LSM Workbench: Converted Data

Field Contents　Change Display　Display Colour Legend

File　FI01_TRANS_FI01_TRANS_FI01_TRANS.lsmw.conv

Row	Structur	Contents		
1	FI01	FI01	FI01	CN 10000051
2	FI01	FI01	FI01	CN 10000052
3	FI01	FI01	FI01	CN 10000053
4	FI01	FI01	FI01	CN 10000054
5	FI01	FI01	FI01	CN 10000055
6	FI01	FI01	FI01	CN 10000056
7	FI01	FI01	FI01	CN 10000057
8	FI01	FI01	FI01	CN 10000058
9	FI01	FI01	FI01	CN 10000059
10	FI01	FI01	FI01	CN 10000050

图 6-111

单击按钮返回 LSMW 处理界面。

步骤十二：创建批处理进程。

选中 Create Batch Input Session 选项，单击按钮，显示如图 6-112 所示。

LSM Workbench: Generate Batch Input Folder

File Name (with Path)	FI01_TRANS_FI01_TRANS_FI01_TRANS.lsmw.conv
Display Trnacts per BI Folder	
Name of Batch Input Folder(s)	FI01_TRANS
User ID	BCUSER
☐ Keep Batch Input Folder(s)?	

图 6-112

单击按钮，进程创建结果如图 6-113 所示。

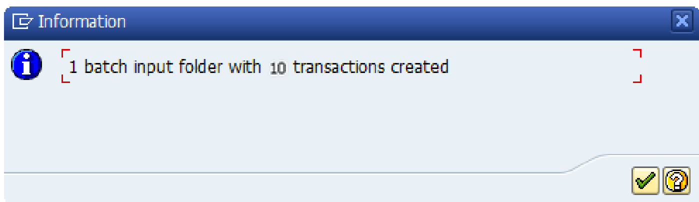


图 6-113

单击✔按钮返回 LSMW 处理界面。

步骤十三：运行批处理进程。

选中 Run Batch Input Session 选项，单击按钮，显示如图 6-114 所示。

Batch Input: Session Overview

Analysis　Process　Statistics　Log　Recording

Selection criteria

Sess.: FI01_TRANS　From:　To:　Created by: *

New　Incorrect　Processed　In Process　In Background　Being Created　Locked

Session name	Sta...	Created By	Date	Time	Creation Program	Lock Date	Authorizat.	Trans.			Screens
FI01_TRANS		BCUSER	2012/12/04	14:49:44	/SAPDMC/SAP_		BCUSER	10	0	0	20

图 6-114

选中该进程并单击 Process 按钮，处理参数选择如图 6–115 所示。

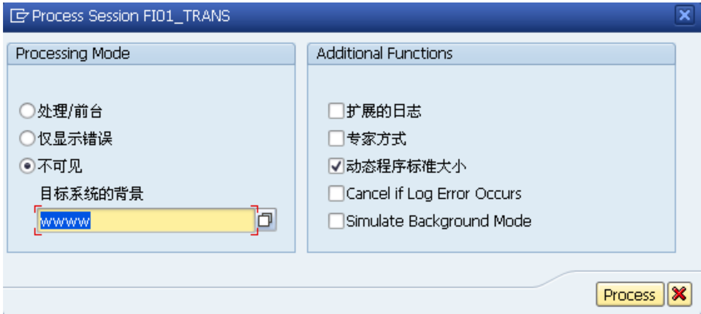


图 6–115

单击 Process 按钮，运行进程。查看进程 Log 结果如图 6–116 所示。

Batch Input Log for Session FI01_TRANS

Choose

Log attributes

Name FI01_TRANS　Queue ID 12120414494426527084　User BCUSER

Created On 2012/12/04　TemSe ID BDCLG442652708402289　Details

Time	Message	Transaction	Index	Modul	Scr...	In...	T	M...	M...
14:55:51	会话 FI01_TRANS 正被用户 BCUSER 在服务器 wwww 的模式 N 中处理		0			0	S	00	300
14:55:53	银行 CN 10000051 被建立	FI01	1	SAPMF02B	0110	2	S	B..	208
14:55:53	事务处理成功	FI01	1			0	S	00	355
14:55:53	银行 CN 10000052 被建立	FI01	2	SAPMF02B	0110	2	S	B..	208
14:55:54	事务处理成功	FI01	2			0	S	00	355
14:55:54	银行 CN 10000053 被建立	FI01	3	SAPMF02B	0110	2	S	B..	208
14:55:54	事务处理成功	FI01	3			0	S	00	355
14:55:54	银行 CN 10000054 被建立	FI01	4	SAPMF02B	0110	2	S	B..	208
14:55:54	事务处理成功	FI01	4			0	S	00	355
14:55:54	银行 CN 10000055 被建立	FI01	5	SAPMF02B	0110	2	S	B..	208
14:55:54	事务处理成功	FI01	5			0	S	00	355

图 6–116

生成数据如图 6–117 所示。

数据浏览器：表格 BNKA 选择条目　　10

检查表格...

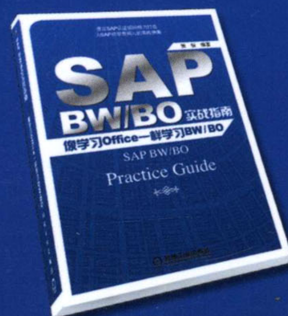
表　：　BNKA

被显示的字段：　13 的 19　固定栏目：　3　列表宽度　0250

客户端	银行国家	银行代码	创建日期	创建者	银行名称	地区	街道
000	CN	10000050	2012/12/04	BCUSER	中国银行	010	北京朝阳区王府井街
000	CN	10000051	2012/12/04	BCUSER	中国银行	010	北京朝阳区鲁迅街
000	CN	10000052	2012/12/04	BCUSER	中国银行	010	北京海淀区长安街
000	CN	10000053	2012/12/04	BCUSER	中国银行	010	北京四临区胡同街
000	CN	10000054	2012/12/04	BCUSER	中国银行	010	北京中山区步行街
000	CN	10000055	2012/12/04	BCUSER	中国银行	010	北京高新区火炬街
000	CN	10000056	2012/12/04	BCUSER	中国银行	010	北京沙河区爱仙街
000	CN	10000057	2012/12/04	BCUSER	中国银行	010	北京千经区人仙街
000	CN	10000058	2012/12/04	BCUSER	中国银行	010	北京维和区鲁班街
000	CN	10000059	2012/12/04	BCUSER	中国银行	010	北京六口区眼光街

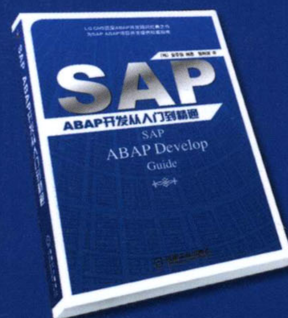
图 6–117

进阶阅读



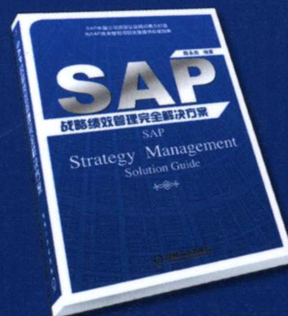
《SAP BW/BO实战指南——像学习Office一样学习BW/BO》

ISBN 978-7-111-37555-5



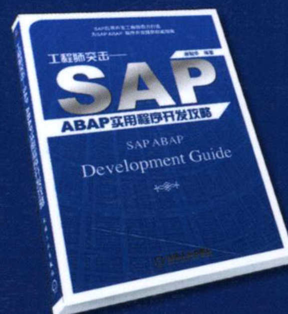
《SAP ABAP开发从入门到精通》

ISBN 978-7-111-41700-2



《SAP 战略绩效管理完全解决方案》

ISBN 978-7-111-30582-8



《工程师突击——SAP ABAP实用程序开发攻略》

ISBN 978-7-111-29891-5

ISBN 978-7-111-50126-8

本书的目标读者群：

- SAP ABAP开发工程师
- SAP 实施顾问
- 企事业单位SAP系统运维工程师
- 各大院校计算机及相关专业师生
- 有志于从事SAP开发行业的爱好者

本书主要介绍了ABAP面向对象编程、ALV列表、接口、增强及ABAP开发人员求职面试的相关技术问题。ABAP面向对象编程部分由浅入深地讲解了面向对象的相关概念，并从实际应用出发，举例说明了ABAP编程所涉及的本地对象和全局对象的编辑实现及应用效果；ALV列表部分全面地介绍了ALV的种类以及各种ALV的实现方式，并详述了面向对象ALV列表及ALV树形列表的编辑实现及应用效果；接口部分系统地介绍了各类接口的原理及实现方法，并详述了IDOC、RFC、BAPI的相关概念及原理；增强部分对增强的升级及不同时期的各代增强的原理、查找办法、实现方式作了详细阐述，并举例说明了每一代增强的编辑实现及应用效果；ABAP面试问题及答案部分提供了较为全面的面试问题集，为求职者打开方便之门。

本书提供了大量配套资源及实例源码，深入剖析了SAP NetWeaver架构的关键技术，是Java等开发人员深入学习SAP系统的必备指南，还适用于SAP技术人员和Java\.NET平台下SAP接口程序的开发人员。

上架指导：SAP/软件开发

51CTO学院推荐

为中华崛起传播智慧

地址：北京市百万庄大街22号
邮政编码：100037

电话服务
服务咨询热线：010-88361066
读者购书热线：010-68326294
010-88379203

网络服务
机工官网：www.cmpbook.com
机工官博：weibo.com/cmp1952
金书网：www.golden-book.com
教育服务网：www.cmpedu.com
封面无防伪标均为盗版


机械工业出版社
微信服务号


计算机分社
微信服务号



策划编辑◎丁诚 / 封面设计◎道乐文化

定价：69.00元